PROYECTO FIN DE CARRERA

Ing. Técnica Informática de Gestión

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA HERRAMIENTA DE GESTIÓN DE REQUISITOS

ALBERTO MAYO SÁNCHEZ

ÍNDICE

1. INTRODUCCIÓN 7
1.1. OBJETIVO 9
1.2. REQUISITOS E INGENIERÍA DE REQUISITOS 10
1.3. LAS HERRAMIENTAS DE GESTIÓN DE REQUISITOS 13
1.3.1. Funcionalidades básicas de estas herramientas 14
2. ANTECEDENTES Y ESTADO DE LA CUESTIÓN 15
2.1. EL ENTORNO DE DESARROLLO E IMPLANTACIÓN 15
2.2. EVALUACIÓN DE HERRAMIENTAS EXISTENTES 17
2.2.1. Requerimientos Analizados para evaluar las herramientas 18
2.2.2. Valoración de las herramientas seleccionadas 19
2.2.3. Interfaz de las herramientas analizadas 22
2.2.4. Conclusiones sobre las herramientas analizadas 25
2.3. NECESIDADES QUE DEBE CUBRIR EL SISTEMA 26
2.3.1. Definición de los requisitos. Estructura. Relaciones. 26
2.3.2. Ciclo de vida de los requisitos 26
2.3.3. Explotación de datos en el sistema de Gestión de Requisitos 27
2.3.4. Relación entre los requisitos y otros posibles productos 27
2.3.5. Otras necesidades e ideas 28
3. ANÁLISIS 29
3.1. ESPECIFICACIÓN DE REQUISITOS 29
3.1.1. Características deseables de una ERS 30
3.1.2. ERS del Sistema Gestión de Requisitos 30
3.2. DIAGRAMA DE CASOS DE USO 34
3.2.1. Explicación diagrama de Casos de Uso 35

3.3. CASOS DE USO 36
3.3.1. LOGIN APLICACIÓN 36
3.3.2. ALTA_REQUISITO 38
3.3.3. EDICIÓN_REQUISITO 41
3.3.4. MODIFICACIÓN DE ATRIBUTOS 42
3.3.5. ASOCIAR DIRECTORIO 43
3.3.6. MODIFICACIÓN_ESTADO 45
3.3.7. MODIFICACIÓN_ESTADO_REQ_SUBREQ 57
3.3.8. MODIFICACION_ESTADO_REQ_LAST_SUBREQ 59
3.3.9. MODIFICACIÓN_ESTADO_REQ_PADRE 62
3.3.10. MOVER_REQUISITO 63
3.3.11. MOVER_REQUISITO_COMP 65
3.3.12. RELACIONAR_REQUISITO 67
3.3.13. SUGERIR_PRIORIDAD 68
3.3.14. MODIFICAR_PRIORIDAD_SUGERIDA 70
3.3.15. FIJAR_PRIORIDAD 72
3.3.16. ASIGNAR_PARTICIPANTES 73
3.3.17. MODIFICAR_VERSIÓN DE DESARROLLO 75
3.3.18. GENERAR_DOC_REQUISITO 77
3.3.19. BÚSQUEDA_REQUISITO 80
3.3.20. BUSQUEDA_ROL 81
3.3.21. BUSQUEDA_ROL_ESTADO 82
3.3.22. CAMBIAR_VISTA_VERSIÓN DE DESARROLLO 84
3.3.23. CAMBIAR_VISTA_JER 85
3.3.24. CAMBIAR_VISTA_ARQ 85

3.4. ATRIBUTOS DE UN REQUISITO 87
3.5. DIAGRAMAS DE FASE DE ANÁLISIS 92
3.5.1. Modelo Entidad Relación 92
3.5.2. Descripción Entidades Modelo E/R 93
3.5.3. Modelo de Clases Conceptual 96
3.5.4. Análisis Capa de Presentación. Patrón MVC 99
4. DISEÑO 101
4.1. REALIZACIÓN DE CASOS DE USO 103
4.2. ARQUITECTURA DEL SISTEMA 105
4.2.1. Aspectos de Arquitectura 105
4.2.2. Diagrama de Arquitectura 107
4.3. DIGRAMA DE ESTADOS.CICLO DE VIDA DEL REQUISITO 108
4.4. SUBSISTEMA DE ACCESO A DATOS 112
4.4.1. Descripción 112
4.4.2. Componentes básicos 113
4.4.3. Detalles de implementación 113
4.4.4. Modelo de clases del subsistema 114
4.4.5. Métodos principales 115
4.5. SUBSISTEMA DE LOGICA DE NEGOCIO EN CLIENTE 116
4.5.1. Descripción 116
4.5.2. Componentes básicos 116
4.5.3. Detalles de implementación 117
4.5.4. Modelo de clases del subsistema 118
4.6. SUBSISTEMA DE LOGICA DE NEGOCIO EN SERVIDOR 120
4.6.1. Descripción 120
4.6.2. Componentes básicos 120
4.6.3. Detalles Implementación 121

4.6.4. Modelo de clases del subsistema 121
4.6.5. Métodos Principales 124
4.6.6. Métodos Principales 125
4.7. SUBSISTEMA DE PRESENTACIÓN 126
4.7.1. Descripción 126
4.7.2. Componentes básicos 126
4.7.3. Detalles de implementación 126
4.7.4. Modelo de clases del subsistema 127
4.7.5. Métodos Principales 129
4.8. LA INTERFAZ GRÁFICA 130
5. ORGANIZACIÓN DEL PROYECTO 133
5.1. CONCEPTO DE PROYECTO 133
5.2. MODELO DEL CICLO DE VIDA DEL SOFTWARE 134
5.2.1. Modelo de Aproximación incremental 134
5.3. PLANIFICACIÓN DEL PROYECTO 136
5.3.1. Diagrama de Gantt 136
5.3.2. Recursos 141
5.3.3. Costes 145
6. PRUEBAS 151
6.1. CONCEPTOS Y DEFINICIONES 151
6.2. DEFINICIÓN CASOS DE PRUEBA 152
7. FUTURAS LÍNEAS DE TRABAJO 182
7.1. CONSIDERACIONES GENERALES 182
7.2. ARQUITECTURA 182
7.3. INTEGRACIÓN CON OTRAS HERRAMIENTAS 183
8. CONCLUSIONES 184
8.1. REFERENTES AL SISTEMA 184
8.2. REFERENTES AL DESARROLLO DEL PROYECTO 185

9. BIBLIOGRAFIA 186
9.1. MEMORIA DEL SISTEMA 186
9.2. IMPLEMENTACIÓN DEL SISTEMA 187

1. INTRODUCCIÓN

En este proyecto se aborda la gestión de requisitos dentro del campo de la Ingeniería del Software, proporcionando una solución de gestión, que facilite la realización de los procesos relacionados con tal compleja actividad.

En los proyectos software se han venido produciendo hechos que influyen negativamente en el desarrollo normal y correcto de los proyectos, desde los primeros tiempos del desarrollo del software:

- Retrasos considerables en cuanto a la planificación
- Falta de productividad
- Elevadas cargas de mantenimiento
- Demandas desfasadas con respecto al producto entregado
- Baja calidad y fiabilidad del producto
- Alta dependencia de los creadores del producto

Hechos como estos condujeron a lo que se denominó la crisis del software. Como solución a la crisis del software surge la "Ingeniería del Software" que es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software. Lo que se pretendía era el tratamiento sistemático de todas las fases del ciclo de vida del software. Era necesario convertir el desarrollo del software en una disciplina de ingeniería de modo que se aplicasen principios y métodos repetibles que permitiesen obtener software de modo rentable.

Para poder aplicar principios y métodos de ingeniería se debe de organizar el proceso de desarrollo del software y sistematizar cada una de las partes y tareas de dicho proceso. Esta formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del software o *ciclo de vida* del desarrollo del software. El ciclo de vida es el periodo de tiempo que comienza cuando se decide desarrollar un producto sistema software, y finaliza cuando se entrega.

Podemos hacer una distinción de las diferentes fases del desarrollo del software, teniendo claro, que no son fases universales, en el sentido de que no siempre se les denomina de la misma forma.

FASES DEL CICLO DE VIDA DEL SOFTWARE:

- Análisis
- Diseño
- Implementación
- Pruebas
- Instalación
- Mantenimiento

El sistema que nos ocupa se apoya en el Proceso Unificado de Software. El Proceso Unificado de Software es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales.

El Proceso Unificado de Software está dirigido por los *casos de uso*. Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Cada caso de uso, está íntimamente relacionado con un requisito, y, radica aquí la importancia de los requisitos del software. Por decirlo de una manera más cercana al lenguaje coloquial, los requisitos software van a ser los cimientos de los sistemas software, con la importancia que esto obviamente conlleva.

Un sistema software se crea para servir a sus usuarios. Por lo tanto, para construir un sistema exitoso se debe conocer qué es lo que quieren y necesitan los usuarios prospectos. El término usuario se refiere no solamente a los usuarios humanos, sino a otros sistemas. En este contexto, el término usuario representa algo o alguien que interactúa con el sistema por desarrollar.

Una especificación funcional tradicional se concentra en responder la pregunta: ¿Qué se supone que el sistema debe hacer? Y ahí juegan un papel determinante, el objeto de estudio en este proyecto, los requisitos software. Otra pregunta a tener en cuenta sería, ¿Qué tiene que hacer el sistema para cada usuario? Estas tres palabras tienen una implicación importante: nos fuerzan a pensar en términos del valor a los diferentes usuarios y no solamente en términos de las funciones que sería bueno que tuviera. Para todo ello, el sistema será más efectivo, si esas funcionalidades que debe de cumplir, que se reflejan en los requisitos, son estudiadas y gestionadas por medio de

una gestión adecuada de los requisitos, no sólo funcionales, que se localizan, realizando un análisis exhaustivo de la especificación del problema a resolver.

En el contexto de las fases del ciclo de vida, una de las fases con mayor importancia e impacto en el desarrollo del sistema, es la fase de **ANÁLISIS**. Dentro del análisis se engloban sub-fases más específicas, entre las cuáles se encuentra el *Análisis de Requisitos*. La clave de esta fase radica en la gestión adecuada de los requisitos encontrados, que debe de cumplir el futuro sistema. Podemos hacernos la pregunta de por qué es un factor clave, y lo es, debido a que el concepto que en última instancia establece la calidad de un sistema software, viene determinado a partir de la concordancia entre los requisitos fijados y la consecución de los mismos. Considerando además que la fase de Gestión de Requisitos se encuentra al comienzo del ciclo de vida, es fácil suponer que todas las no conformidades que se encuentren al principio del ciclo, supondrán menor coste que las que se encuentren una vez que se ha avanzado en el desarrollo. El objetivo final siempre es que el sistema a desarrollar realice lo que desde un principio queríamos que realizara, y para ello es fundamental el rigor y la exhaustividad en toda la fase de análisis.

1.1. OBJETIVO

El Objetivo con la realización de este proyecto, es la **creación** de un **sistema** de gestión de requisitos, que **facilite** la propia labor de la **gestión de requisitos en el desarrollo de productos software**, tarea perteneciente a la Ingeniería de Requisitos, y que resulta fundamental durante la fase de Análisis, dentro del ciclo de vida del software. La gestión de requisitos es una actividad difícil de ejecutar, por varios factores, citando entre ellos, la dificultad de mantenimiento, actualización, y a la vez los desencuentros entre equipos de desarrollo que se producen en ella. El sistema pretende **acabar con ineficiencias introduciendo una metodología englobada en el sistema**, facilitando la formalización del proceso, y logrando que la gestión de requisitos sea un conjunto de tareas eficaces que ayuden en el desarrollo del software. Se busca poder llegar a convertirse en una **alternativa a los sistemas existentes en el mercado**, pudiendo ser una herramienta parametrizada de acuerdo a las necesidades y demás metodologías utilizadas por los posibles usuarios de la misma.

1.2. REQUISITOS E INGENIERÍA DE REQUISITOS

La Ingeniería de Requisitos es una disciplina perteneciente a la Ingeniería del Software. Tiene un objetivo claro. Ese objetivo es la comprensión de las necesidades que debe cumplir un sistema a desarrollar y que son identificadas por los futuros usuarios del sistema. La ingeniería de requisitos está compuesta por un conjunto de tareas. Esas tareas permitirán identificar las necesidades anteriormente citadas y transformarlas en los requerimientos que debe de cumplir el sistema que se desarrollen, identificando adicionalmente y de manera concisa:

- Alcance
- Dirección
- Propósito

Se muestra una definición de la Ingeniería de Requisitos:

> INGENIERÍA DE REQUISITOS: *Es un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requerimientos del sistema siguiendo un determinado estándar.* BIBLIOGRAFÍA [1]

La ingeniería de requisitos está presente no sólo en el desarrollo de proyectos software sino también en numerosas prácticas de ingeniería, y la importancia de la misma resulta vital en muchos casos, como puede ser en el desarrollo del software.

El proceso de Ingeniería de Requisitos se adapta a los distintos modelos del ciclo de vida del software, siendo el modelo elegido, una consideración no influyente en la generación del documento de requisitos. El producto final del proceso es un documento de requisitos, y ese es el objetivo de la Ingeniería de Requisitos. En este proceso de creación del documento de requisitos, participan recursos y esos recursos se apoyan generalmente en herramientas y técnicas que ayudan a que el proceso sea eficiente.

La especificación de requisitos software es uno de los productos que se obtienen dentro de la fase análisis, siendo el objetivo de dicha fase es el siguiente:

> OBJETIVO FASE DE ANÁLISIS: *La producción de un documento de especificación de requisitos, que describa lo que el futuro sistema debe hacer, pero no cómo debe hacerlo.* BIBLIOGRAFÍA [9]

La especificación de requisitos es generalmente una de las primeras tareas que se llevan a cabo, y el objetivo es realizar una primera identificación de los requisitos del sistema. No todos los requisitos identificados en la tarea de especificación de requisitos, son finalmente implementados, esto quiere decir, que un requisito hasta que es desarrollado en el caso de que lo sea, pasa por diversos estados. Es necesario por tanto gestionar esas modificaciones, y de ahí nace el propósito de la gestión de requisitos, poder controlar esos cambios sobre los requisitos identificados en una primera instancia con el fin de evitar pérdida de información relevante. La gestión de requisitos es un proceso crítico, y puede afectar al desarrollo del sistema que se lleva a cabo.

Se efectúa una aproximación a una definición de Gestión de Requisitos:

> **GESTIÓN DE REQUISITOS**: *La Gestión de Requisitos en Ingeniería de Requisitos, es el proceso encargado de la identificación, asignación y seguimiento de los requisitos, incluyendo la interfaz, verificación, modificación y control del estado a lo largo del ciclo de vida.* BIBLIOGRAFÍA [3]

Algunas de las actividades que se llevan a cabo en la Gestión de Requisitos, son las presentadas a continuación, teniendo en cuenta que no existe un orden estrictamente secuencial entre ellas, pudiendo realizarse en varias ocasiones algunas de ellas.

- Extracción de requisitos: Permite descubrir cuáles son los requisitos del sistema. Se definen y se refinan para obtener una descripción lo más precisa posible. Intervienen desde clientes a proveedores. Se realizan entrevistas para recoger información.

- Análisis de requisitos: Se realiza un análisis de los requisitos que se identifican en la extracción, buscando encontrar inconsistencias. Se lleva a cabo una especificación de requisitos. El diagrama de flujo de datos es una técnica que facilita la realización de esta actividad.

- Validación y verificación de requisitos: Tras efectuar un documento de especificación de requisitos, la verificación busca comprobar si la especificación realizada sobre cada uno de los requisitos, atiende a las necesidades que se establecieron en el comienzo del proyecto. El cliente debe colaborar activamente en esta actividad.

- Gestión de cambios: Se consigue reflejar todos los cambios que se puedan llevar a cabo sobre las especificaciones iniciales. Cualquier cambio afecta directamente a los requisitos identificados, y es necesario por tanto notificar dichos cambios.

Para más información y detalle sobre la gestión de requisitos, se pueden consultar las siguientes referencias:

- BIBLIOGRAFIA [3],[7],[10] [11], [12]

1.3. LAS HERRAMIENTAS DE GESTIÓN DE REQUISITOS

En cualquier proyecto, la gestión de los requisitos es una tarea ardua y en la mayoría de las veces siempre resulta compleja. Las herramientas de Gestión de Requisitos surgen para simplificar estas tareas. El uso de estas herramientas proporciona una serie de ventajas evidentes y que permiten no sólo una previsible mejora en los procesos, y consecuentemente su influencia en la calidad final del producto, sino una serie de ventajas adicionales, y no tomadas en cuenta siempre por los equipos de desarrollo software:

El uso de herramientas de gestión de requisitos permite [BIBLIOGRAFIA 3]:

- Ahorrar en costes de desarrollo minimizando el impacto de errores.
- Mejorar la calidad mediante un adecuado análisis y gestión de los requisitos.
- Facilitar la reutilización.
- Aumentar la productividad.
- Reducir las no-conformidades del sistema.
- Controlar y administrar más fácilmente la especificación.
- Cumplir con estándares de calidad.
- Proporcionar un repositorio de especificaciones.
- Centralizar la información del problema.
- Especificar sistemas de forma estructurada y gráfica.
- Obtener la trazabilidad de la especificación.

En el siguiente punto se presentan las funcionalidades que ofrecen las herramientas de Gestión de Requisitos para cumplir con las ventajas descritas anteriormente.

1.3.1. Funcionalidades básicas de estas herramientas

Las características básicas que debe cumplir una herramienta para ser considerada como herramienta de Gestión de Requisitos [BIBLIOGRAFIA 3] son:

- Identificación de requisitos globales y detallados
- Asignación a un destino y clasificación de requisitos.
- Agrupación de requisitos, revisión, e identificación de requisitos iniciales
- Interfaz de datos básicos.

Las funciones que además debe cumplir la herramienta son identificación de requisitos, revisión y edición, trazabilidad desde su origen y generación de informes. Además debe permitir analizar el impacto de un cambio, identificando los requisitos afectados.

La función de gestión, requerida por las herramientas consiste en la recopilación de métricas y supervisión de la estabilidad de los requerimientos a través del control del cambio. Además como funciones añadidas las herramientas pueden aportar capacidades de simulación, o capacidades de integración con otras herramientas (herramientas de pruebas, planes de test etc.)

2. ANTECEDENTES Y ESTADO DE LA CUESTIÓN

El objetivo de este apartado dentro del proyecto, es situar el proyecto en un entorno, describiendo la situación anterior al desarrollo del mismo, y el mundo de alrededor. Es importante, ayuda, aunque de un modo bastante simplificador, a apreciar el trabajo realizado y a apreciar, con algo de mayor claridad, las primeras aportaciones, y novedades que se aportan sobre los precedentes, los cuáles se analizarán también en este apartado.

2.1. EL ENTORNO DE DESARROLLO E IMPLANTACIÓN

El Sistema de Gestión de Requisitos se puede plantear como una necesidad para una organización implicada en el desarrollo de software, que requiere de una herramienta de gestión, parametrizada de acuerdo a sus características, que le ayude en el proceso de Gestión de Requisitos dentro del Proceso de Desarrollo del De Software, y esté acorde a las metodologías que se utilicen.

Una de las cuestiones que provoca más problemas durante el desarrollo del software y que supone un esfuerzo mayor de mantenimiento, actualización, etc. es la Gestión de Requisitos. Se puede convertir en una tarea muy difícil de ejecutar de manera correcta y bastante conflictiva por los desencuentros de los distintos equipos de trabajo. Considerando que es una de las etapas más importantes del ciclo de vida del software, es necesario encontrar en esta fase un conjunto de tareas eficaces que apoyen en el desarrollo, en vez de dificultarlo constantemente. Así pues como acción correctiva es posible introducir una metodología englobada en un sistema que permita gestionar de manera eficaz los requisitos de los proyectos.

El Sistema de Gestión de Requisitos, podría interactuar con otras aplicaciones que puedan existir, ya sean herramientas de resolución de incidencias, dónde las incidencias podrían asociarse siguiendo el camino hacia el análisis, con alguno de los requisitos encontrados previamente, o bien tomando el camino opuesto, con una herramienta de pruebas, como puede ser un generador de casos de test, dónde cada caso también podría estar relacionado con un requisito encontrado en la fase de análisis.

Como objetivo general del desarrollo del Sistema de Gestión de Requisitos, se tomaría, el mejorar una actividad fundamental del Análisis previo al desarrollo de un sistema cualquiera, como es la gestión de los requisitos, para minimizar errores en las

posteriores fases del desarrollo, y así adaptarse al máximo posible a las necesidades que quiere resolver el sistema a construir.

Objetivo final : MEJORA EN LA CALIDAD

A continuación, se presentan una serie de procesos que una organización, que pudiera responder a la necesidad de un Sistema de Gestión de Requisitos como el que es objeto de este proyecto, pudiera llevar en su día a día, en el desarrollo de aplicaciones:

- Se recogen los requisitos de distintas fuentes cómo los dpto. de Proyectos, de los clientes…etc.
- Existe una plantilla aceptada por los diferentes departamentos dónde se describen los requisitos
- Mediante una reunión se realiza una puesta en común de los requisitos, entre los departamentos afectados
- Se envía un correo electrónico a todas las personas implicadas, y se les requiere *feedback* para aprobar o no esos requisitos.
- Tras la aprobación, mediante una nueva reunión, se fija, que se va hacer en cada uno de los requisitos, comenzando así la fase de Análisis.
- Durante la evolución del producto por las sucesivas fases del desarrollo, no se almacena información relevante sobre los requisitos implementados ,entre los departamentos partícipes en esas fases.

Si algunos de estos procesos se llevan a cabo en una organización dónde se desarrolle software, estamos ante una organización que debería de mejorar su proceso de gestión de requisitos, y por tanto, potencial beneficiario del Sistema de Gestión de Requisitos.

2.2. EVALUACIÓN DE HERRAMIENTAS EXISTENTES

El apartado de la evaluación de las herramientas de Gestión de Requisitos existentes en el mercado, corresponde al Estado de la Cuestión.

¿Por qué evaluar las herramientas existentes en el mercado?

Una característica general de la Ingeniería del Software, que es independiente del área de aplicación, es la fase de definición. Durante la definición, quienes desarrollan el software intentan identificar qué información ha de ser procesada, qué funciones se desea que cumpla el sistema, y por tanto ha de identificarse los requisitos clave del sistema y del software.

Una organización, que, por ejemplo, realiza la labor inicial de recogida, de almacenamiento y administración, mediante hojas de cálculo, puede plantearse la necesidad de modificar ese proceso de recogida. Como sistema de recogida inicial de requisitos puede parecer correcto hasta el momento en el que se valora la necesidad de tener un sistema que centralice toda esta información, que permita gestionarla y relacionarla con otros productos de trabajo. Y además y lo más importante, que permita mantener una trazabilidad y seguimiento de los requisitos, de modo que en cada momento se pueda saber cómo está evolucionando un desarrollo. Esto es lo que justifica la utilización de una herramienta capaz de gestionar los requisitos software asociados a un determinado desarrollo.

Ante esta situación, una organización con la gestión de requisitos de sus desarrollos, de la manera que se comenta, tiene, si quiere mejorar en el proceso, cambiar su metodología en esta parte del ciclo de vida. Las opciones para llevar a cabo los propósitos de mejora son claras:

- ✓ Utilizar una herramienta disponible en el mercado.
- ✓ Invertir en el desarrollo de una herramienta que satisfaga las propias necesidades de quienes desarrollan en una organización.

En primer lugar se debe llevar a cabo un análisis de los productos disponibles en el mercado, estableciendo un proceso de evaluación de las distintas herramientas de Gestión de Requisitos que pueden resolver las necesidades encontradas. En el siguiente apartado se muestra dicha evaluación.

2.2.1. Requerimientos Analizados para evaluar las herramientas

A continuación se describen los criterios que se estiman que debieran cumplir cada una de las herramientas que se proceden a analizar, junto a las características técnicas y funcionales de cada producto.

- ***Requerimientos tecnológicos***
    - Modelo de datos flexible, capacidad para adaptar estructuras existentes.
    - Interfaz API disponible para ser manejado desde herramientas propias.
    - Escalabilidad en los datos, soporte a grandes volúmenes de datos.
    - Acceso remoto al sistema a través de Internet/Intranet.
    - Gestión de roles, usuarios, departamentos.
    - Sistema de consultas e informes parametrizable.
    - Integración con Office.
    - Sistema de auditoría por roles.
    - Usabilidad.

- ***Requerimientos funcionales***
    - Integración con gestión de configuración.
    - Edición similar a herramientas disponibles de gestión de errores y pruebas.
    - Gestión de incidencias similar la forma de gestión de errores y de pruebas.
    - Manejo de distintas versiones de producto.
    - Búsquedas.
    - Gestión documental.
    - Compatibilidad e integración con otras herramientas.
    - Gestión de requerimientos por arquitectura y por jerarquía.

- Posibilidad de comunicación de responsables de un requisito por la herramienta.
- Permitir importación de requisitos.
- Asociar documentación.
- Trazabilidad de requisitos.
- Gestión de cambios.

- ***Oros requerimientos***

  - Soporte
  - Mantenimiento
  - Fiabilidad
  - Formación

2.2.2. Valoración de las herramientas seleccionadas

Las herramientas existentes en el mercado, que pudieran efectuar una gestión de los requisitos óptima, que sirven de base al desarrollo del Sistema de Gestión de Requisitos son las siguientes:

- CALIBER RM
- DOORS
- REQUISITE PRO
- IRQA

En la tabla de la página siguiente, se valoran las herramientas, de acuerdo a una serie de características, relacionadas con los requisitos funcionales y tecnológicos encontrados anteriormente:

	CALIBER RM	DOORS	REQUISITE PRO	IRQA
			×	✓
Modelo de datos	-	-	×	✓
Interfaz API	✓	✓	×	✓
Escalabilidad	-	✓	✓	✓
Acceso remoto	Sólo lectura	✓	-	-
Gestión de roles	-	-	-	-
Consultas e informes	✓	-	-	✓
Office	✓	-	-	✓
Auditoría	-	-	-	-
Usabilidad	-	✓	-	✓
Gestión de configuración	-	-	-	✓
Edición similar	-	-	-	×
Gestión incidencias	-	-	-	×
Varias versiones	-	-	-	-
Búsquedas	-	-	-	✓
Gestión documental	✓	-	-	✓

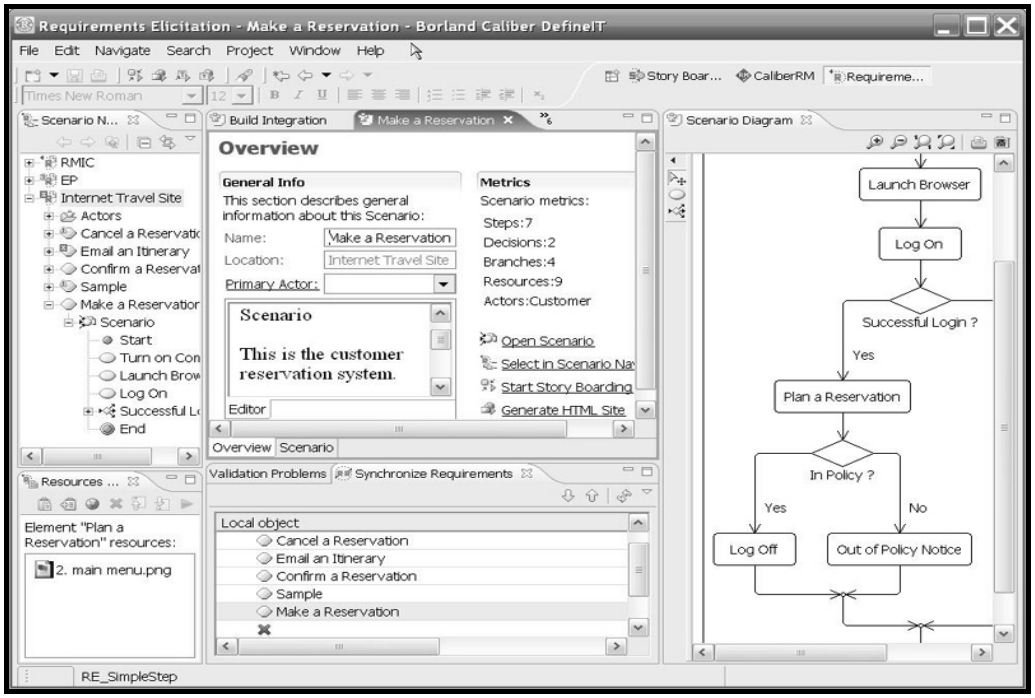
	CALIBER RM	DOORS	REQUISITE PRO	IRQA
Integración herramientas	-	Muy costosa	×	-
Gestión por árbol	✓	✓	✓	✓
Comunicación por herramienta	✓	-	×	-
Importación de requisitos	-	-	-	-
Asociar documentación	✓	No 1 por req.	-	-
Trazabilidad	✓	-	×	-
Gestión de cambios	✓	-	×	-
Soporte	×	✓	✓	✓
Mantenimiento	×	✓	✓	✓
Formación	×	✓	✓	✓
Fiabilidad	×	✓	✓	✓
Tiempo de proyecto	-	-	Alto	Alto
Precio	Medio	Medio	Alto	Alto

2.2.3. Interfaz de las herramientas analizadas

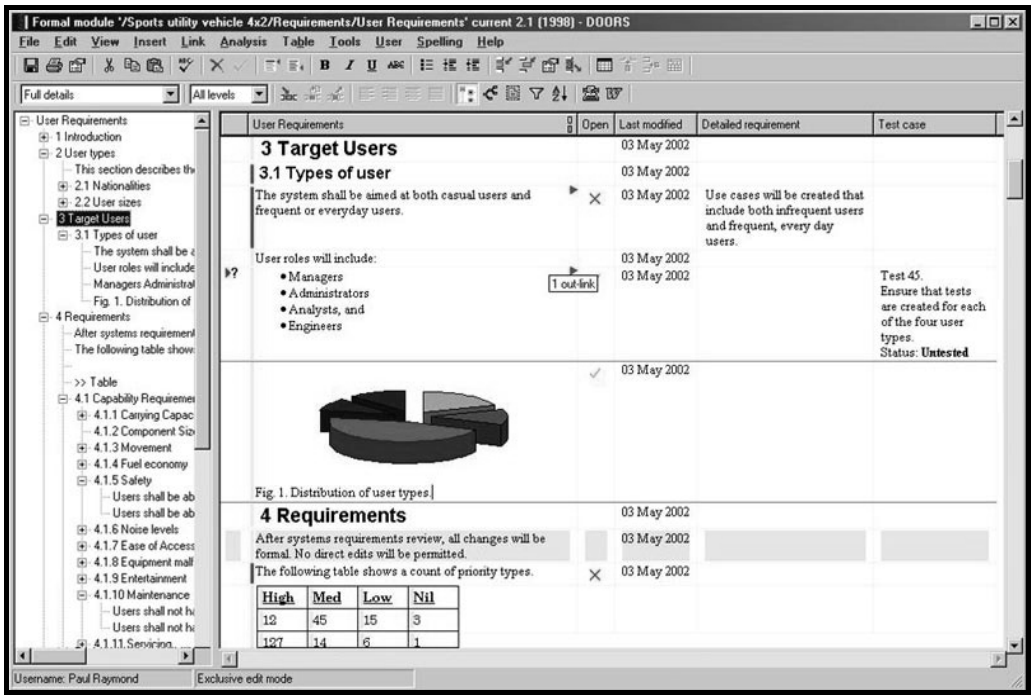
Es importante, en las tomas de decisiones relacionadas con acabar con la disyuntiva de si conviene utilizar una herramienta nueva, el factor aprendizaje del usuario final de esa herramienta. Evidentemente la interfaz de un sistema, afecta al tiempo de aprendizaje que conlleva el uso de un sistema nuevo.

En este apartado se muestran algunas capturas de las interfaces de los cuatro productos de gestión de requisitos analizados:

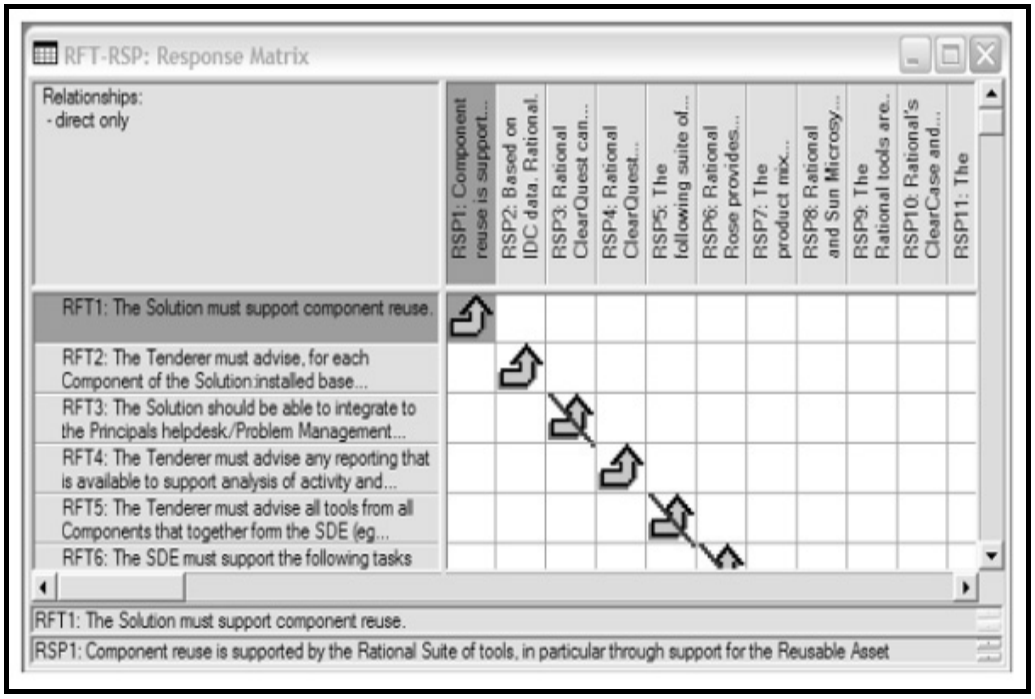
- CALIBER RM – Construcción de un escenario



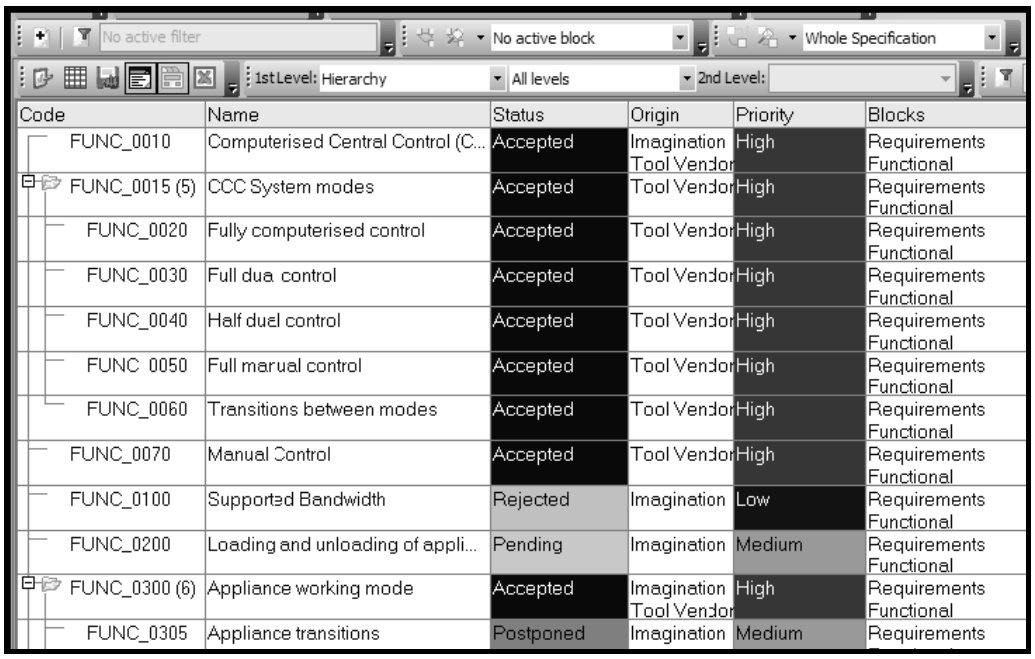
❖ DOORS – Vista de organización de requisitos



❖ REQUISITE PRO – Matriz de trazabilidad de requisitos



- IRQA – Vista de manejo de requisitos



Code	Name	Status	Origin	Priority	Blocks
FUNC_0010	Computerised Central Control (C...	Accepted	Imagination Tool Vendor	High	Requirements Functional
FUNC_0015 (5)	CCC System modes	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0020	Fully computerised control	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0030	Full dua control	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0040	Half duel control	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0050	Full marual control	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0060	Transitions between modes	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0070	Manual Control	Accepted	Tool Vendor	High	Requirements Functional
FUNC_0100	Supported Bandwidth	Rejected	Imagination	Low	Requirements Functional
FUNC_0200	Loading and unloading of appli...	Pending	Imagination	Medium	Requirements Functional
FUNC_0300 (6)	Appliance working mode	Accepted	Imagination Tool Vendor	High	Requirements Functional
FUNC_0305	Appliance transitions	Postponed	Imagination	Medium	Requirements

2.2.4. Conclusiones sobre las herramientas analizadas

- **Caliber RM** :

Imposibilidad de mantener una comunicación directa con la empresa distribuidora y la posible lentitud en recibir respuestas a los problemas que puedan surgir durante la implantación de la herramienta.

- **Doors**:

Excesiva complejidad en la posible integración con las herramientas propias de un departamento de I+D de gestión de errores y gestión de pruebas. El manejo del lenguaje propietario demanda un alto nivel de aprendizaje.

- **Requisite PRO**:

Se centra demasiado en los documentos más que en la propia Gestión de los Requisitos y la dificultad de integrar el resto de herramientas propias con esta solución. No está dotada de todas las características funcionales que debiera.

- **IRqA**:

Herramienta que más se ajusta a los requisitos funcionales y tecnológicos expuestos. Coste mayor que el de otras herramientas.

2.3. NECESIDADES QUE DEBE CUBRIR EL SISTEMA

Dentro de este apartado, se redactan las necesidades que el sistema a desarrollar debe de satisfacer, y estas necesidades paralelamente se convierten en ideas para el análisis que se realiza a posteriori.

2.3.1. Definición de los requisitos. Estructura. Relaciones.

I. Los requisitos deben organizarse en una estructura FLEXIBLE, que permitan relaciones jerárquicas, como de otros tipos,

II. En la jerarquía de requisitos podemos distinguir, al menos, los requisitos estratégicos, tácticos y de implementación. Son posibles más niveles de organización.

III. En el alta de nuevos requisitos éstos deben clasificarse por funcionalidad.

IV. Los requisitos deben versionarse.

V. Se debe permitir la entrada de prioridades en un requisito por parte de los participantes que tengan permiso para ello.

2.3.2. Ciclo de vida de los requisitos

I. El sistema debe permitir el alta de los requisitos en el momento y lugar que se capten.

II. El sistema debe permitir la definición de un grupo de participantes y garantizar que los cambios en los requisitos y en su documentación asociada llegan correctamente a todos ellos.

III. Los requisitos deben circular por un workflow de estados, y los cambios deben comunicarse a todos los participantes.

IV. Debería de existir un mecanismo adicional, y en ocasiones alternativo, a las reuniones para discutir requisitos, que permita documentar más fácilmente las decisiones y los puntos de vista de los participantes.

V. No deberían de aceptarse ni gestionarse requisitos no dados de alta en el sistema, allí dónde se implante, para conseguir una implantación exitosa del mismo.

VI. Posible distinción entre requisito y mejora:

a. **Mejora** *es una nueva funcionalidad, con un impacto muy controlado en el producto y que implica poco esfuerzo de desarrollo y pruebas.*

b. **Requisito** *es toda propuesta de nueva funcionalidad más importante que una mejora.*

VII. Tanto los requisitos como su documentación deben incluir un control de cambios que sea lo más automático posible

VIII. Los requisitos no se borran. El rechazo debería reflejarse, indicando las causas del mismo.

2.3.3. Explotación de datos en el sistema de Gestión de Requisitos

I. El sistema debe ser capaz de generar informes con la información relativa a cada requisito.

II. El sistema debe facilitar la organización del trabajo de los participantes en los requisitos (qué temas debo validar, qué cambios debo ver en la definición o en la documentación, etc.).

III. En la medida de lo posible las exportaciones que pudieran realizarse al Office de los requisitos deben respetar, o parecerse a, los formatos y plantillas que se usen antes de la realización del sistema, para evitar más carga de aprendizaje.

2.3.4. Relación entre los requisitos y otros posibles productos

I. El sistema debería enlazar claramente los requisitos y el resto de productos de trabajo realizados durante el desarrollo del software, para así garantizar una trazabilidad lo más completa posible.

II. Los requisitos deben estar relacionados con su documentación de análisis, diseño e implementación.

III. Debería ser sencillo establecer una relación entre un requisito y su esfuerzo de desarrollo, ya sea estimado, para los aún no realizados, o real, para los ya implementados.

2.3.5. Otras necesidades e ideas

I. Fundamental: Usabilidad del sistema. Debe ser un sistema muy usable, para evitar en primer lugar, grandes periodos de aprendizaje, y la posibilidad de que no se implante exitosamente.

II. Sistema para todos. Que existan muy diversos tipos de usuarios, para que la herramienta se extienda en su uso.

Para más información y detalle sobre las diferentes herramientas mostradas, se pueden consultar las siguientes referencias:

- BIBLIOGRAFIA [3],[6]

3. ANÁLISIS

Cuando hablamos del término Análisis, en la coyuntura de la realización de proyectos, estamos hablando de una fase esencial en el desarrollo de cualquier proyecto, y quizá la fase más importante, y no sólo en el desarrollo de proyectos software. Todo lo que quede fuera del Análisis, son aspectos perdidos en las siguientes fases del proyecto, de ahí lo importante que resulta esta fase en general, y volviendo a lo comentado en la introducción de este proyecto, el análisis y la gestión de requisitos, forman parte de ella. El producto que obtendremos tras la realización de esta fase viene siendo un documento dónde se especifica qué debe hacer nuestro sistema, pero no cómo debe hacerlo, labor que será desarrollada en la fase de diseño del sistema.

3.1. ESPECIFICACIÓN DE REQUISITOS

La Especificación de Requisitos del Software, que responde a las siglas de ERS, se puede definir como la documentación de los requisitos esenciales:

- Funciones
- Rendimiento
- Diseño
- Restricciones
- Atributos

La ERS debe de ser eficaz, y para ello debe de tener ciertas características que la hagan accesible al equipo de proyecto y facilite su utilización para el trabajo técnico, y a la vez para la interrelación con el futuro usuario del sistema. No olvidemos que una de las fuentes de extracción de requisitos puede ser el futuro usuario. La ERS es un producto fundamental para el correcto desarrollo del software. La ERS es al desarrollo de software, lo que los planos representan en los aspectos de una casa en la arquitectura (no su estructura de construcción).

Debe abordar la DESCRIPCIÓN de lo que vamos a construir, no el cómo, ni tampoco el cuándo, y en ella se deben de describir los requisitos del software pero no requisitos innecesarios, ni tampoco ningún detalle del posible diseño.

3.1.1. Características deseables de una ERS

Las características deseables de toda ERS, son las siguientes:

- No ambigua
- Completa
- Fácil de verificar
- Fácil de modificar
- Clasificada por importancia
- De fácil utilización durante las demás fases

3.1.2. ERS del Sistema Gestión de Requisitos

En la ERS se muestran los requisitos encontrados, que debe de cumplir nuestros sistema una vez desarrollado. Se define la composición de un requisito, las jerarquías e interrelaciones con otros componentes.

Cada requisito encontrado tendrá asociado un ID, bajo la siguiente nomenclatura:

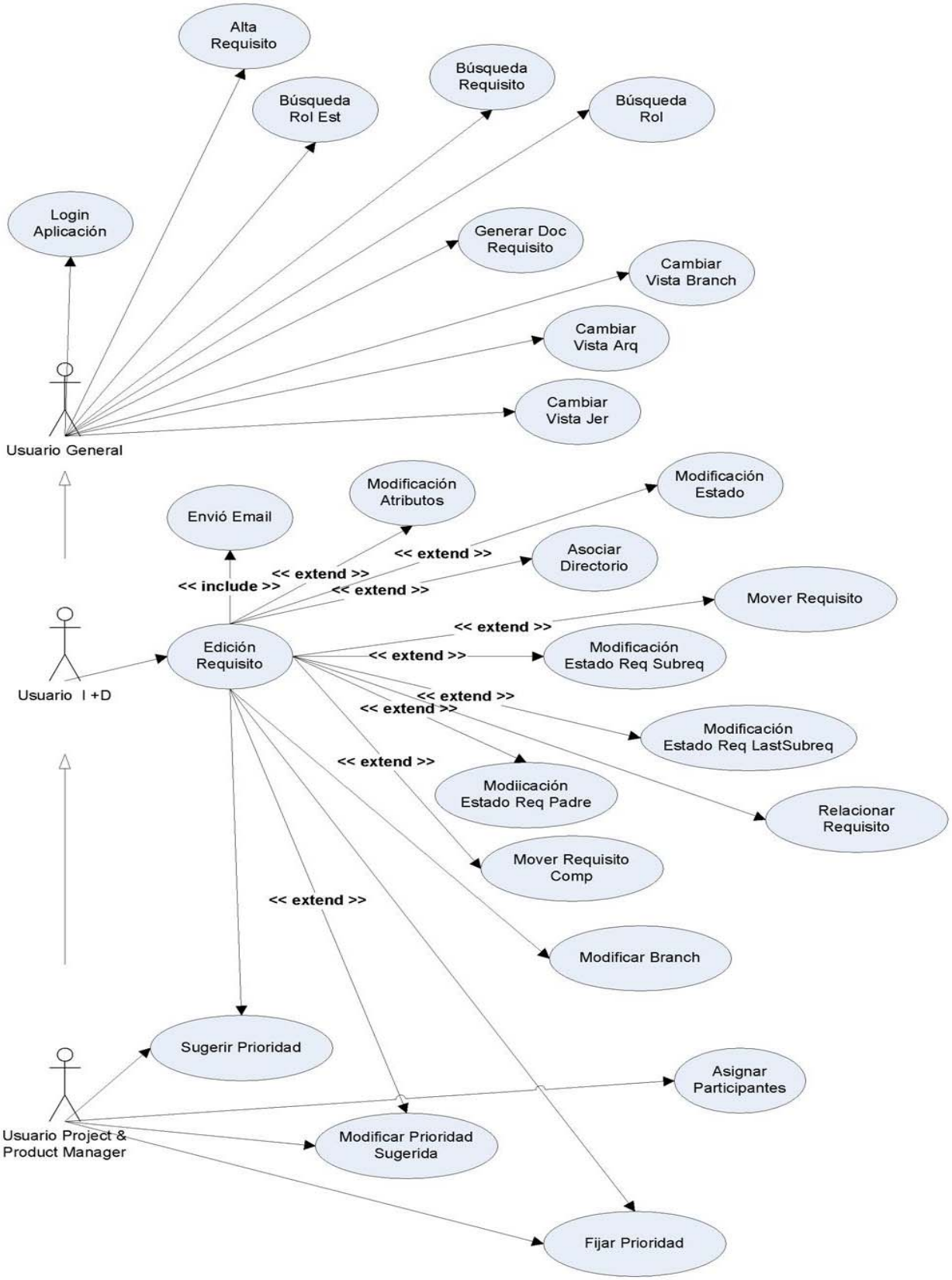
R i, dónde i indica el número de orden del requisito

ID	DESCRIPCIÓN
R1	En el Sistema de Gestión de Requisitos (SGR) se pueden añadir todos los requisitos que correspondan tanto a desarrollos tecnológicos como funcionales. De entrada los requisitos se añaden a una versión de producto no definida, con unos datos mínimos entre los que se encuentra una prioridad, y posteriormente el *product manager* de cada área decidirá su clasificación, estado y revisará las características del requisito.
R2	Cada requisito puede sufrir modificaciones a lo largo de su vida en el sistema. El sistema guarda los cambios de dichos requisitos y se encarga de que los participantes en el mismo estén informados de cada modificación.
R2.1	Los requisitos sufrirán una evolución a través de distintos estados: abierto, aprobado, en desarrollo, en codificación, cerrado, denegado, obsoleto.
R.2.2	Los requisitos podrán cambiar de ubicación, dentro de las diferentes versiones del producto, es decir podrán "nacer" para una versión y a posteriori cambiar y desarrollarse en otra.
R.2.3	Cada modificación de un requisito generará un email , informando a cada participante en el requisito de la modificación efectuada..
R3	El SGR por cada requisito podrá generar una salida que será un documento formal de requisitos, mostrando la trazabilidad del requisito. Dicho informe permitirá manejar la información por escrito al igual que se manejan actualmente los documentos de requisitos.
R4	La gestión de requisitos, durante su ciclo de vida en el sistema, se basará en roles con permisos asociados para cambios de estado en los requisitos. Estarán asignados a personas diferentes en función del contenido de cada requisito concreto.
R5	El sistema permitirá visualizar los requisitos por componentes y/o por funcionalidad, y realizar distintas consultas de búsqueda.
R6	El SGR debe permitir relacionar requisitos en dos formas, jerárquica y horizontal. Estas relaciones permitirán tener una trazabilidad completa de los requisitos.

R6.1	Relación jerárquica entre requisitos, de modo que podemos tener distintos número de niveles según el requisito en el que nos encontremos: si es un requisito de alto nivel, o de bajo nivel.
R6.2	Relación horizontal entre requisitos para mostrar extensiones de requisitos, consideradas como nuevos requisitos, o relaciones de impacto en el caso de requisitos de configuración, los cuales tienen su origen en requisitos funcionales
R7	Los requisitos podrán y deberán asociarse a otros productos de trabajo, quedando constancia de ello relacionándolos con los respectivos documentos que lo prueben.
R7.1	Asociación con planes de test: - A partir de una estructura de requisitos de bajo nivel (específicos/implementación) se puede generar automáticamente un esqueleto de Plan de pruebas. A partir de ahí mantener la correspondencia entre requisitos y casos de test garantiza que no hay carencias ni en un lado ni en el otro. - Contando con ésta asociación y las relaciones entre requisitos es más fácil definir las fases de regresión de pruebas.
R7.2	Asociación con documentos de análisis y diseño. Permite seguir todo el ciclo de vida del requisito mediante los documentos pero con toda la información centralizada, y así también mantenerlos actualizados. Manteniendo éstos enlaces sería sencillo enviar correos a los participantes en un requisito sobre cuando se actualiza un documento (basta con revisar su fecha de actualización periódicamente y avisar cuando cambie).
R7.3	Asociación con documentos de usuario. Cualquier cambio en un requisito o cualquier nueva funcionalidad derivada de él, que pueda impactar en la documentación, será considerada gracias a esta asociación.

R7.4	Asociación con tareas del Project. Un requisito siempre vive en el contexto de un *project*, si él no está asociado directamente a un mpp, lo estará indirectamente al que tenga asociado su requisito padre, abuelo... El *project* inicial debería generarse y actualizarse a partir de la estructura de requisitos definida en la aplicación. Es posible que el *project* contenga tareas no asociadas a requisito pero no a la inversa. Idealmente las fechas de estimación del requisito deben alimentarse en *project* y desde ahí leerse en la aplicación.
R9	Los requisitos deben clasificarse en: - Funcionales - No funcionales: tecnológicos, de integración, de testeo. El sistema puede incorporar algunos chequeos en este sentido para evitar que se olvide hacer esta distinción. Los requisitos no funcionales serán bastante concretos y relacionados directamente con la funcionalidad.

3.2. DIAGRAMA DE CASOS DE USO



3.2.1. Explicación diagrama de Casos de Uso

Un diagrama de Casos de Uso, o también conocido como diagrama de contexto, es una notación gráfica para representar Casos de Uso. Hay que diferenciar entre Caso de Uso, y diagrama de Casos de Uso. Aunque los dos términos están relacionados, el diagrama representa la naturaleza de un conjunto de casos de uso, mientras que el Caso de Uso, es una explicación detallada.

Los casos de uso están representados por elipses y los actores están representados por las figuras humanas. Los actores son una especie de rol, un usuario humano u otra entidad externa que puede jugar varios papeles o roles.

Para el sistema de Gestión de Requisitos, definimos en una primera instancia tres tipos de actores del sistema. En primer lugar un actor general, que tendrá asociados ciertos casos de uso. Un actor I+D heredado del usuario general, un actor Project & Product Manager heredado del usuario I+D.

3.3. CASOS DE USO

3.3.1. LOGIN APLICACIÓN

IDENTIFICACIÓN CASO DE USO				
ID	LOGIN_APLICACION			
NOMBRE	Login a la aplicación			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona.			
DESCRIPCION	El usuario realiza login en la aplicación de gestión de requisitos, introduciendo usuario y contraseña.			
DISPARADOR	El usuario arranca la aplicación.			
PRECONDICIONES				
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El usuario se conecta a la aplicación.			
	POSTCONDICION FALLIDA			
	El sistema deniega el acceso a la aplicación.			
PRIORIDAD	Muy alta			
FRECUENCIA DE USO	Muy Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario arranca la aplicación de gestión de requisitos e introduce sus credenciales.		
			2	El sistema recoge datos de usuario.
			3	El sistema permite conexión.
			4	El sistema asigna permisos según tipo de usuario.
	5	El usuario obtiene acceso a la aplicación con asignación de permisos realizada.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema no obtiene datos de usuario, porque son erróneos o inexistentes: usuario no existe, ó password es incorrecto
			3	El sistema deniega acceso informando de la situación al usuario.
	4	El usuario no puede acceder a la aplicación.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema no permite conexión, por factores ajenos

			4	El sistema deniega acceso informando de la situación al usuario.
	5	El usuario no puede acceder a la aplicación.		
EXCEPCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.2. ALTA_REQUISITO

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">ALTA_REQUISITO</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Creación y registro de un requisito en el sistema.</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario crea un nuevo requisito en la aplicación.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor añade requisito.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">Login en la aplicación y permisos asociados. Estado inicial del requisito: CHEQUEADO o ABIERTO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se crea correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la creación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Muy alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="7">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario introduce datos del requisito:
- Título, abreviatura
- Tipo
- Descripción
- Estado
- Prioridad inicial
- Versión de desarrollo: general (limbo)
- Categoría</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema asocia un id de requisito (temporal hasta grabación de requisito), asocia versión, y asocia creador de requisito y valida los datos de entrada.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema valida si el estado es un estado permitido para el usuario.</td></tr>
<tr><td></td><td></td><td>4</td><td>Cuando el estado = CHEQUEADO, el sistema chequea la asignación de responsable en desarrollo y del responsable en QA, así como del resto de participantes, incluye el caso de uso ASIGNAR_PARTICIPANTES.</td></tr>
<tr><td>5</td><td>El usuario asigna responsable en desarrollo y responsable en QA, así como</td><td></td><td></td></tr>
</table>

		participantes, y además si el estado es CHEQUEADO, da valor determinado a la versión de desarrollo y no lo añade al general.		
			6	El sistema graba el nuevo requisito.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema produce error en la validación de los datos.
			3	El sistema informa al usuario del dato introducido que es erróneo.
	4	El usuario vuelve a introducir datos y el caso continúa en el punto 2 del camino normal.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema impide al usuario asignar el estado elegido debido al tipo de usuario.
			4	El sistema informa al usuario de la situación y le solicita un nuevo estado
	5	El usuario asigna un nuevo estado y el caso continúa en el punto 3 del camino normal.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			4a	Cuando el estado CHEQUEADO, el sistema no chequea responsable de desarrollo, ni de QA, ni participantes y el caso de uso continúa en el punto 6 del camino normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			6a	El sistema da error (por falta de datos o por datos incorrectos) al grabar el requisito e informa al usuario.
			7	El requisito no se da de alta y acaba el caso de uso.

FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema produce error en la validación de los datos.
			3	El sistema informa al usuario del dato introducido que es erróneo.
	4	El usuario vuelve a introducir datos y el caso continúa en el punto 2 del camino normal.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema impide al usuario asignar el estado elegido debido al tipo de usuario.
			4	El sistema informa al usuario de la situación y le solicita un nuevo estado
	5	El usuario asigna un nuevo estado y el caso continúa en el punto 3 del camino normal.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			4a	Cuando el estado CHEQUEADO, el sistema no chequea responsable de desarrollo, ni de QA, ni participantes y el caso de uso continúa en el punto 6 del camino normal.
EXCEPCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.3. EDICIÓN_REQUISITO

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">EDICIÓN_REQUISITO</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Edición de un requisito en el sistema.</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario edita algún aspecto de un requisito en la aplicación.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor abre un requisito.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">Login en la aplicación y permisos asociados. Estado inicial del requisito: CHEQUEADO o ABIERTO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se edita correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la edición del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="4">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario introduce datos a modificar del requisito</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema inserta los cambios introducidos al requisito. Los cambios se describen en cada uno de los casos de uso que extienden este caso de uso:
-MODIFICACIÓN_ATRIBUTOS
-ASOCIAR_DIRECTORIO
-MODIFICACIÓN_ESTADO
-MODIFICACIÓN_ESTADO_ REQ_SUBREQ
-MODIFICAR_ESTADO_REQ_ LASTSUBREQ
-MODIFICAR_ESTADO_REQ_ PADRE
-MOVER_REQUISITO
-MOVER_REQUISITO_COMP
-RELACIONAR_REQUISITO
-AÑADIR_PRIORIDAD_ SUGERIDA
-MODIFICAR_PRIORIDAD_ SUGERIDA
-FIJAR_PRIORIDAD
-ASIGNAR_PARTICIPANTES
-MODIFICAR_VERSIÓN DE DESARROLLO</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema graba el requisito con las modificaciones y acaba el caso de uso.</td></tr>
</table>

3.3.4. MODIFICACIÓN DE ATRIBUTOS

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">MODIFICACION_DE_ATRIBUTOS</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Edición atributos de un requisito en el sistema.</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario modifica atributos de un requisito en la aplicación.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor abre requisito.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se crea correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la creación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Muy Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario introduce datos que desea modificar del requisito:
- Título, abreviatura
- Tipo.
- Descripción.
- Categoría.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea si el usuario tiene permisos para cambiar atributos.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema cambia automáticamente la versión del requisito.</td></tr>
<tr><td></td><td></td><td>4</td><td>El sistema graba la información.</td></tr>
<tr><td></td><td></td><td>5</td><td>Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>2a</td><td>El usuario no tiene permisos para modificar el requisito, el sistema no puede realizar ninguna acción.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>3a</td><td>No se modifica ningún atributo luego se termina el caso de uso y no se cambia la versión.</td></tr>
</table>

FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			4a	El sistema da error al grabar el requisito e informa al usuario.
			5	No se modifican los atributos del requisito.
EXCEPCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.5. ASOCIAR DIRECTORIO

IDENTIFICACIÓN CASO DE USO				
ID	ASOCIAR_DIRECTORIO			
NOMBRE	Asociar directorio que incluye documento a relacionar con el requisito.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: desarrollo, calidad, product manager, project manager, editor, analista/diseñador, documentalista…			
DESCRIPCION	El usuario asocia un directorio.			
DISPARADOR	El actor modifica requisito.			
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito se modifica correctamente.			
	POSTCONDICION FALLIDA			
	Error en la modificación del requisito.			
PRIORIDAD	Media			
FRECUENCIA DE USO	Media			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario añade un directorio al requisito.		
			2	El sistema chequea si el usuario tiene permisos para añadir directorios. Si el usuario es el responsable en desarrollo, responsable en QA, product, ó documentalista, entonces

				podrá añadir documentos.
			3	El sistema graba la información.
			4	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El usuario no tiene permisos para añadir directorios al requisito, el sistema no puede realizar ninguna acción.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema da error al grabar el requisito e informa al usuario.
				No se le añade el nuevo directorio al requisito y acaba el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			5a	Da error al enviar el email a los participantes, con lo cual no se produce la comunicación de la modificación.
EXCEPCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.6. MODIFICACIÓN_ESTADO

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">MODIFICACIÓN_ESTADO</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Modificar estado del requisito</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuarios de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario modifica el estado del requisito.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor modifica requisito.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Muy Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="4">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CHEQUEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea si el usuario tiene permisos para modificar estado a CHEQUEADO. Si el usuario es el responsable en desarrollo, responsable en QA, product manager o validador, entonces podrá poner a CHEQUEADO el requisito.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita al usuario el responsable de desarrollo y el de QA, así como el resto de participantes; incluye caso de uso ASIGNAR_PARTICIPANTES. Además el sistema solicita versión de desarrollo en la que se va a desarrollar el requisito.
El sistema solicita también fijar categoría al requisito si éste no la tiene fijada ya.</td></tr>
</table>

	4	El usuario añade responsables de desarrollo y de QA, así como el resto de participantes. El usuario añade versión de desarrollo al requisito. El usuario añade categoría al requisito.		
			5	El sistema graba la información.
			6	Se graba el cambio en el histórico Información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema detecta que usuario no tiene permisos para chequear requisito. Da un mensaje de aviso al usuario y termina el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			5a	El sistema da error al grabar el requisito e informa al usuario.
			6	No se modifica el estado del requisito y acaba el caso de uso.
FLUJO NORMAL DE EJECUCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado DESECHADO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese ABIERTO, CHEQUEADO, CODIFICADO, TESTEADO ó ANALIZADO, y permite cambiar el requisito a estado DESECHADO.
			3	El sistema solicita introducir causa de cambio de estado a DESECHADO, que se introducirá en el comment to status.
	4	El usuario añade causa de DESECHADO.		
			5	El sistema graba la información y el caso de uso continúa en el camino 7 del caso normal.

<table>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado DESECHADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2a</td><td>El sistema chequea que el estado anterior del requisito fuese ABIERTO, CHEQUEADO CODIFICADO, TESTEADO ó ANALIZADO, y como es un estado distinto no permite cambiar el requisito a DESECHADO y advierte al usuario con un mensaje. Acaba el caso de uso.</td></tr>
<tr><td rowspan="6">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado DESECHADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese ABIERTO, CHEQUEADO CODIFICADO, TESTEADO ó ANALIZADO, y permite cambiar el requisito a estado DESECHADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita introducir causa de cambio de estado a DESECHADO, para incluirla en el comment to status.</td></tr>
<tr><td>4b</td><td>El usuario no introduce causa.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema no permite grabar información.</td></tr>
<tr><td rowspan="5">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado DESECHADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese ABIERTO, CHEQUEADO CODIFICADO, TESTEADO ó ANALIZADO, y permite cambiar el requisito a estado DESECHADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita introducir causa de cambio de estado a DESECHADO,</td></tr>
</table>

				para incluirla en el comment to status.
	4	El usuario añade causa de DESECHADO.		
			5a	Se produce error al grabar el requisito en db e informa al usuario y acaba el caso de uso.
FLUJO NORMAL DE EJECUCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado ANALIZADO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, y permite cambiar el requisito a estado ANALIZADO.
			3	El sistema solicita directorios de análisis y diseño, y fecha de beta.
	4	El usuario introduce directorios de análisis y diseño, tipo del requisito si procede, y fecha de beta.		
			5	El sistema graba la información y el caso de uso continúa en el camino 7 del caso normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado ANALIZADO el requisito.		
			2a	El sistema comprueba que el estado anterior no era CHEQUEADO, no permite cambiar a este estado el requisito y advierte al usuario con un mensaje acabando el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado ANALIZADO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, y permite cambiar el requisito a estado ANALIZADO.

<table>
<tr><td></td><td></td><td></td><td>3a</td><td>El sistema da error al solicitar directorios, o fecha, advierte al usuario con un mensaje de error y no se permite modificar estado del requisito, acabando el caso de uso.</td></tr>
<tr><td rowspan="6">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado ANALIZADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, y permite cambiar el requisito a estado ANALIZADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita directorios de análisis y diseño y fecha de beta.</td></tr>
<tr><td>4a</td><td>El usuario no introduce datos obligatorios y no se produce el cambio de estado del requisito acabando el caso de uso.</td><td></td><td></td></tr>
<tr><td rowspan="7">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado ANALIZADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, y permite cambiar el requisito a estado ANALIZADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita directorios de análisis y diseño, revisión del tipo de requisito, fecha de estimación de CODIFICADO.</td></tr>
<tr><td>4</td><td>El usuario introduce directorios de análisis y diseño, tipo del requisito si procede, y fecha de estimación de CODIFICADO y TESTEADO.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5a</td><td>Se produce error al grabar la información, no se modifica el estado del requisito y no se graba la información.</td></tr>
</table>

<table>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CODIFICADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese ANALIZADO, y permite cambiar el requisito a estado CODIFICADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita directorio de CODIFICADO.</td></tr>
<tr><td>4</td><td>El usuario introduce directorios de CODIFICADO y opcionalmente relaciona el requisito con otros requisitos de manera horizontal (esto lo puede realizar en cualquier momento del ciclo).</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema graba la información y el caso de uso continúa en el camino 7 del caso normal.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CODIFICADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2a</td><td>El sistema comprueba que el estado anterior no era ANALIZADO, no permite cambiar a este estado el requisito y advierte al usuario con un mensaje acabando el caso de uso.</td></tr>
<tr><td rowspan="5">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CODIFICADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese ANALIZADO, y permite cambiar el requisito a estado CODIFICADO.</td></tr>
<tr><td></td><td></td><td>3a</td><td>El sistema da error al solicitar directorio, advierte al usuario con un mensaje de error y no se permite modificar estado del requisito,</td></tr>
</table>

				acabando el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado CODIFICADO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese ANALIZADO, y permite cambiar el requisito a estado CODIFICADO.
			3	El sistema solicita directorio de CODIFICADO.
	4	El usuario introduce directorio de CODIFICADO, y relaciona requisito de manera horizontal con un requisito que sea padre o hijo de sí mismo.		
			5a	Se produce error al grabar la información, y no graba el cambio de estado del requisito y advierte con un mensaje al usuario, acabando el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado CODIFICADO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese ANALIZADO, y permite cambiar el requisito a estado CODIFICADO.
			3	El sistema solicita directorio de CODIFICADO.
	4	El usuario introduce directorio de CODIFICADO y opcionalmente relaciona el requisito con otros requisitos de manera horizontal.		
			5b	El sistema da error al grabar la información, no se modifica el estado del requisito y no se graba la información.

<table>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema solicita directorio de plan de test.</td></tr>
<tr><td>4</td><td>El usuario introduce directorio de TESTEADO, además opcionalmente relaciona el requisito con otros requisitos de manera horizontal.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema graba la información y el caso de uso continúa en el camino 7 del caso normal.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema comprueba que el estado anterior no era CODIFICADO, no permite cambiar a este estado el requisito y advierte al usuario con un mensaje acabando el caso de uso.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2a</td><td>El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
</table>

			2b	El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.
			3a	El sistema da error al solicitar directorio, advierte al usuario con un mensaje de error y no se permite modificar estado del requisito, acabando el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado TESTEADO el requisito.		
			2b	El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.
			3	El sistema solicita directorio de plan de test.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado TESTEADO el requisito.		
			2b	El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.
			3	El sistema solicita directorio de plan de test.
	4	El usuario introduce directorio de TESTEADO, y relaciona requisito de manera horizontal con un requisito que sea padre o hijo de sí mismo.		
			5a	Se produce error al grabar la información, y no graba el cambio de estado del requisito y advierte con un mensaje al usuario, acabando el caso de uso.

<table>
<tr><td rowspan="6">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1d</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.</td></tr>
<tr><td></td><td></td><td>4</td><td>El sistema solicita directorio de plan de test.</td></tr>
<tr><td>5</td><td>El usuario introduce directorio de TESTEADO, además opcionalmente relaciona el requisito con otros requisitos de manera horizontal.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>6a</td><td>Se produce error al grabar la información, no se modifica el estado del requisito y no se graba la información.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1d</td><td>El usuario cambia a estado TESTEADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea que el estado anterior del requisito fuese CODIFICADO, y permite cambiar el requisito a estado TESTEADO.</td></tr>
<tr><td rowspan="4">FLUJO NORMAL DE EJECUCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CERRADO el requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema comprueba que el estado anterior del requisito no era TESTEADO y advierte al usuario del error en el cambio de estado y no permite grabar ningún cambio de estado.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia a estado CERRADO el requisito.</td><td></td><td></td></tr>
</table>

			2	El sistema chequea que el estado anterior del requisito fuese TESTEADO, y permite cambiar el requisito a estado CERRADO.
			3a	Se produce error al grabar la información y no se graba nada del cambio de estado.
FLUJO NORMAL DE EJECUCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado APARTADO		
			2	El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, ANALIZADO ó CODIFICADO, y permite cambiar el requisito a estado APARTADO.
			3	El sistema graba la información y el caso de uso continúa en el camino 7 del caso normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado APARTADO.		
			2a	El sistema comprueba que el estado anterior del requisito no era CHEQUEADO, ni ANALIZADO, ni CODIFICADO y advierte al usuario del error en el cambio de estado y no permite grabar ningún cambio de estado.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado APARTADO.		
			2	El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, ANALIZADO ó CODIFICADO, y permite cambiar el requisito a estado APARTADO.
			3a	Se produce error al grabar la información y no se graba nada del cambio de estado.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado APARTADO.		
			2	El sistema chequea que el estado anterior del requisito fuese

				CHEQUEADO, ANALIZADO ó CODIFICADO, y permite cambiar el requisito a estado APARTADO.
			3a	Se produce error al grabar la información y no se graba nada del cambio de estado.
FLUJO NORMAL DE EJECUCION	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia a estado ABIERTO el requisito.		
			2a	El sistema comprueba que el estado anterior no era CHEQUEADO, ó bien que el usuario no es product o project, no permite cambiar a este estado el requisito y advierte al usuario con un mensaje acabando el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1g	El usuario cambia a estado ABIERTO el requisito.		
			2	El sistema chequea que el estado anterior del requisito fuese CHEQUEADO, y permite cambiar el requisito a estado ABIERTO, siempre y cuando el usuario sea product o project.
			3a	Se produce error al grabar la información, no se modifica el estado del requisito y no se graba la información.
EXCEPCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.7. MODIFICACIÓN_ESTADO_REQ_SUBREQ

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">MODIFICACIÓN_ESTADO_REQ_SUBREQ</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Modificar estado de algunos hijos no producen modificaciones en el padre</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona: desarrollo, calidad, product manager, Project manager, editor, analista/diseñador, documentalista.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario modifica el estado de algunos subrequisitos a CERRADO pero no todos los de un padre, luego el padre conserva el estado a CHEQUEADO y el usuario tiene que mover el requisito padre a otra versión de desarrollo.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor modifica requisito que tiene subrequisitos</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia estado de un subrequisito a CERRADO.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema evalúa si el usuario tiene permisos para modificar estado a CERRADO. (Los permisos se asignarán en el login y se podrán consultar durante la sesión).</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema informa al usuario de que hay algunos subrequisitos hermanos de este subrequisito que están en estado CHEQUEADO, luego padre va a continuar a CHEQUEADO.</td></tr>
<tr><td>4</td><td>El usuario replica la rama de requisito padre a otra versión de desarrollo destino dejando el estado a CHEQUEADO y</td><td></td><td></td></tr>
</table>

		llevándose la subestructura de subrequisitos a CHEQUEADO.		
			5	El sistema cambia en la versión de desarrollo origen el estado del requisito padre y de su subestrutura a CERRADO.
			6	El sistema graba la información.
			7	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema detecta que usuario no tiene permisos para cerrar requisito. Da un mensaje de aviso al usuario y termina el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que todos los subrequisitos hermanos están a CERRADO, el caso de uso continúa en el paso 5 del camino normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	4	El usuario no mueve la rama del requisito padre a otra versión de desarrollo.		
			5	El sistema advierte con un mensaje de que no se pueden cerrar los subrequisitos deseados y acaba el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			5a	El sistema da error al cambiar el requisito padre y los hijos a CERRADO en el versión de desarrollo origen, advierte con un mensaje y acaba el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			6a	Se produce error al grabar la información, y advierte con un mensaje al usuario. Acaba el caso de uso.

EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.8. MODIFICACION_ESTADO_REQ_LAST_SUBREQ

IDENTIFICACIÓN CASO DE USO				
ID	MODIFICACIÓN_ESTADO_REQ_LASTSUBREQ			
NOMBRE	Modificar estado de último sub-requisito hijo produce cambios en el requisito padre.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuarios de tipo persona: desarrollo, calidad, Product manager, Project manager, editor, validador, analista/diseñador.			
DESCRIPCION	El usuario modifica el estado del último requisito hijo que esté a un estado distinto de los demás al mismo que los demás. El padre automáticamente cambiará a dicho estado.			
DISPARADOR	El actor modifica sub-requisito			
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito padre se modifica correctamente.			
	POSTCONDICION FALLIDA			
	Error en la modificación del requisito padre.			
PRIORIDAD	Alta			
FRECUENCIA DE USO	Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario cambia estado de un sub-requisito a CERRADO.		
			2	El sistema evalúa si el usuario tiene permisos para modificar estado a CERRADO. (Los permisos se asignarán en el login y se podrán consultar durante la sesión).
			3	El sistema informa al usuario de que hay algunos sub-requisitos hermanos

				de este sub-requisito que están en estado CHEQUEADO, luego padre va a continuar a CHEQUEADO.
	4	El usuario replica la rama de requisito padre a otra versión de desarrollo destino dejando el estado a CHEQUEADO y llevándose la subestructura de sub-requisitos a CHEQUEADO.		
			5	El sistema cambia en la versión de desarrollo origen el estado del requisito padre y de su sub-estructura a CERRADO.
			6	El sistema graba la información.
			7	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema detecta que usuario no tiene permisos para cambiar estado de requisito. Da un mensaje de aviso al usuario y termina el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que no tiene que cambiar ningún requisito padre, continúa el caso de uso en el punto 6 del camino normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	4a	El usuario no acepta el cambio.		
			5	El sistema modifica sólo el requisito actual, y advierte con un mensaje de que los padres no se van a ver afectados. El caso continúa en el punto 6 del camino normal.

FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			5a	El sistema da error al cambiar el requisito padre, modifica el hijo sólo y advierte con un mensaje y acaba el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			6a	El sistema da error al grabar la información, y advierte con un mensaje al usuario. Acaba el caso de uso.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.9. MODIFICACIÓN_ESTADO_REQ_PADRE

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">MODIFICACIÓN_ESTADO_REQ_PADRE</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Modificar estado de algunos hijos no modifica padre</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuarios de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario modifica el estado de un requisito padre, con requisitos hijos.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor modifica requisito que tiene sub-requisitos</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia estado de un requisito padre a CERRADO.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema evalúa si el usuario tiene permisos para modificar estado a CERRADO.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema informa al usuario de que hay algunos sub-requisitos hermanos de este sub-requisito que están en otros estados.</td></tr>
<tr><td>4</td><td>El usuario replica que el requisito padre cambia de estado y por tanto que sus sub-requisitos también.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema cambia el estado del requisito padre y a su subestructura</td></tr>
</table>

				también.
			6	El sistema graba la información.
			7	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema detecta que usuario no tiene permisos para cerrar requisito. Da un mensaje de aviso al usuario y termina el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que todos los sub-requisitos hermanos están a CERRADO, el caso de uso continúa en el paso 5 del camino normal.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			6a	Se produce error al grabar la información, y advierte con un mensaje al usuario. Acaba el caso de uso.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.10. MOVER_REQUISITO

IDENTIFICACIÓN CASO DE USO	
ID	MOVER_REQUISITO
NOMBRE	Cambiar requisito padre de un requisito
DEFINICIÓN CASO DE USO	
ACTOR	Usuarios de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.

<table>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario modifica el estado de un requisito padre, con requisitos hijos.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor modifica requisito que tiene subrequisitos</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="5">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario cambia el requisito padre de un requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea si el usuario tiene permisos para cambiar el requisito padre a un requisito. (Los permisos se asignarán en el login y se podrán consultar durante la sesión). Si el usuario es el responsable en desarrollo, responsable en QA, product, entonces podrá cambiar el requisito padre.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema graba la información.</td></tr>
<tr><td></td><td></td><td>4</td><td>Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>2a</td><td>El usuario no tiene permisos para cambiar el requisito padre al requisito, el sistema no puede realizar ninguna acción.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>3a</td><td>El sistema da error al grabar el requisito e informa al usuario.</td></tr>
</table>

			4	No se cambia el padre del requisito y se queda en la misma posición que tuviese en el árbol.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.11. MOVER_REQUISITO_COMP

IDENTIFICACIÓN CASO DE USO				
ID	MOVER_REQUISITO_COMP			
NOMBRE	Mover requisito de componente, desde la vista de arquitectura de componentes.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.			
DESCRIPCION	El usuario mueve un requisito de un componente a otro.			
DISPARADOR	El actor modifica requisito.			
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito padre se modifica correctamente.			
	POSTCONDICION FALLIDA			
	Error en la modificación del requisito padre.			
PRIORIDAD	Media			
FRECUENCIA DE USO	Baja			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario está en la vista de arquitectura de componentes y mueve el requisito de un componente de arquitectura a otro.		

			2	El sistema chequea si el usuario tiene permisos para cambiar el requisito de componente. (Los permisos se asignarán en el login y se podrán consultar durante la sesión). Si el usuario es el responsable en desarrollo, responsable en QA, product, entonces podrá cambiar el requisito de componente.
			3	El sistema chequea que el estado del requisito es ABIERTO, CHEQUEADO, ó ANALIZADO.
			4	El sistema graba la información.
			5	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El usuario no tiene permisos para cambiar de componente el requisito, se advierte con un mensaje de error al usuario y no se realiza el cambio.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que el estado del requisito es CODIFICADO, TESTEADO, CERRADO, DESECHADO o APARTADO, se advierte dando un mensaje de error al usuario de que no se puede realizar ningún cambio.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			4a	Se produce error a la hora de grabar el requisito, así que no se produce la grabación y aparece un mensaje de error que advierte al usuario de la circunstancia.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	

NOTAS	

3.3.12. RELACIONAR_REQUISITO

IDENTIFICACIÓN CASO DE USO				
ID	RELACIONAR_REQUISITO			
NOMBRE	Relacionar requisitos con otros requisitos.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.			
DESCRIPCION	El usuario mueve un requisito de un componente a otro.			
DISPARADOR	El actor modifica requisito.			
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito padre se modifica correctamente.			
	POSTCONDICION FALLIDA			
	Error en la modificación del requisito padre.			
PRIORIDAD	Media			
FRECUENCIA DE USO	Baja			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario relaciona un requisito con otro.		
			2	El sistema chequea si el usuario tiene permisos para añadir una relación a un requisito. (Los permisos se asignarán en el login y se podrán consultar durante la sesión). Si el usuario es el responsable en desarrollo, responsable en QA, product, entonces podrá añadir una relación.
			3	El sistema graba la información.
			4	Se graba en el histórico el cambio la

				información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El usuario no tiene permisos para añadir una relación al requisito, el sistema no puede realizar ninguna acción.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	Se produce error al grabar el requisito e informa al usuario.
			4	No se añade la relación al requisito.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			4a	Se produce error a la hora de grabar el requisito, así que no se produce la grabación y aparece un mensaje de error que advierte al usuario de la circunstancia.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.13. SUGERIR_PRIORIDAD

IDENTIFICACIÓN CASO DE USO	
ID	SUGERIR_PRIORIDAD
NOMBRE	Sugerir prioridad de implementación.
DEFINICIÓN CASO DE USO	
ACTOR	Usuario de tipo persona: desarrollo, calidad, product manager, Project manager, editor, validador, analista/diseñador.
DESCRIPCION	El usuario mueve un requisito de un componente a otro.
DISPARADOR	El actor modifica requisito.
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4

<table>
<tr><td></td><td colspan="4">EDICIÓN_REQUISITO.
El usuario tiene permisos de alto nivel.
El requisito se encuentra en el estado CHEQUEADO o ANALIZADO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="5">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario intenta sugerir una prioridad.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea si el usuario tiene permisos para proponer prioridades de implementación.

Si el usuario es uno de los descritos como Actor en el caso de uso entonces podrá añadir prioridades.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema graba la información.</td></tr>
<tr><td></td><td></td><td>4</td><td>Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>2a</td><td>El usuario no tiene permisos para proponer prioridades de implementación, el sistema no realiza ninguna acción de cambio.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>3a</td><td>El sistema da error al grabar el requisito e informa al usuario.</td></tr>
<tr><td></td><td></td><td>4</td><td>No se añade la nueva prioridad y acaba el caso de uso.</td></tr>
<tr><td rowspan="3">EXCEPCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

NOTAS	

3.3.14. MODIFICAR_PRIORIDAD_SUGERIDA

IDENTIFICACIÓN CASO DE USO				
ID	MODIFICAR_PRIORIDAD_SUGERIDA			
NOMBRE	Cambiar la prioridad del requisito.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: Product Manager, Project Manager			
DESCRIPCION	El usuario cambia su prioridad de implementación del requisito.			
DISPARADOR	El actor modifica requisito.			
PRECONDICIONES	El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO. El usuario tiene permisos de alto nivel. El requisito se encuentra en el estado CHEQUEADO o ANALIZADO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito se modifica correctamente.			
	POSTCONDICION FALLIDA			
	Error en la modificación del requisito.			
PRIORIDAD	Alta			
FRECUENCIA DE USO	Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario intenta cambiar su prioridad.		
			2	El sistema chequea si el usuario tiene permisos para proponer prioridades de implementación. El sistema chequea si la prioridad que desea modificar el usuario es la que él había asignado previamente, o si no es la suya. Si el usuario es uno de los descritos como Actor en el caso de uso entonces podrá modificar prioridades.

			3	El sistema graba la información.
			4	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	La prioridad que se desea modificar no es la perteneciente al usuario conectado, el sistema no da la posibilidad de modificar dicha prioridad.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema da error al grabar el requisito e informa al usuario.
			4	No se añade la prioridad modificada y acaba el caso de uso.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.15. FIJAR_PRIORIDAD

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">FIJAR_PRIORIDAD</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Fijar la prioridad del requisito.</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona: Product Manager</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario fija una prioridad de implementación del requisito, intentando decidir una prioridad que equilibre el contexto del proyecto con las prioridades sugeridas.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor modifica requisito.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4">El usuario está modificando un requisito en el contexto del caso de uso 3.4 EDICIÓN_REQUISITO.
El usuario tiene permisos de alto nivel.
El requisito se encuentra en el estado CHEQUEADO o ANALIZADO.</td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">El requisito se modifica correctamente.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4">Error en la modificación del requisito.</td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Alta</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="5">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario intenta fijar la prioridad.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema chequea si el usuario tiene permisos para fijar la prioridad de desarrollo. Si el usuario es uno de los descritos como Actor en el caso de uso entonces podrá fijar la prioridad.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema graba la información.</td></tr>
<tr><td></td><td></td><td>4</td><td>Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.</td></tr>
<tr><td rowspan="3">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>2a</td><td>El usuario no tiene permisos para fijar la prioridad, el sistema no le da la</td></tr>
</table>

				posibilidad de hacerlo.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	Se produce error al grabar el requisito e informa al usuario.
			4	No se añade la prioridad modificada y acaba el caso de uso.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.16. ASIGNAR_PARTICIPANTES

IDENTIFICACIÓN CASO DE USO				
ID	FIJAR_PRIORIDAD			
NOMBRE	Fijar la prioridad del requisito.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: Product Manager			
DESCRIPCION	El sistema pide que se añadan los participantes del requisito, tras haber colocado el creador el estado inicial			
DISPARADOR	El creador ha colocado estado.			
PRECONDICIONES	Está en el ámbito del caso de uso ALTA_REQUISITO y también en MODIFICAR_ESTADO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito tiene a sus responsables en BBDD.			
	POSTCONDICION FALLIDA			
	Error al asignar responsable.			
PRIORIDAD	Alta			
FRECUENCIA DE USO	Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario asigna participantes del requisito.		
			2	El sistema chequea si los miembros escogidos pertenecen a departamentos

				de I+D, además el sistema chequea que los participantes genéricos no sean los mismos que los asignados como responsables de QA y de desarrollo.
			3	El sistema permite grabar.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	2a	El usuario añade sólo un responsable y no añade ningún otro participante.		
			3	El sistema no permite grabar, e informaría de la situación.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	2b	El usuario añade el mismo responsable en ambos departamentos y añade el resto de participantes.		
			3	El sistema no permite grabar, e informaría de la situación.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que los miembros escogidos en cada departamento o en alguno de ellos no pertenecen al departamento de I+D.
			4	El sistema no permite grabar, e informaría de la situación.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema comprueba que los miembros escogidos en cada departamento o en alguno de ellos son los mismos que los responsables.

			4	El sistema no permite grabar, e informaría de la situación.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.17. MODIFICAR_VERSIÓN DE DESARROLLO

IDENTIFICACIÓN CASO DE USO				
ID	MODIFICAR_VERSIÓN DE DESARROLLO			
NOMBRE	Modificación de la versión de desarrollo a la que está asignado un requisito.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona: Product manager.			
DESCRIPCION	El usuario modifica la versión de desarrollo a la que está asignado un requisito.			
DISPARADOR	El usuario modifica versión de desarrollo.			
PRECONDICIONES	Está en el ámbito del caso de uso MODIFICAR_REQUISITO.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El requisito cambia de versión de desarrollo			
	POSTCONDICION FALLIDA			
	Error al asignar nueva versión de desarrollo			
PRIORIDAD	Alta			
FRECUENCIA DE USO	Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	Usuario modifica la versión de desarrollo de un requisito al editarlo.		
			2	El sistema chequea si el usuario tiene permisos para cambiar versión de desarrollo. Si el usuario es uno de los descritos como Actor en el caso de uso entonces podrá fijar la prioridad.

			3	El sistema graba la información.
			4	Se graba en el histórico el cambio la información antigua, la nueva, el autor y la fecha.
			5	El sistema muestra en la vista por versión de desarrollo, si se filtra por la versión de desarrollo escogida, el requisito recién modificado.
			6	El sistema deja de mostrar en la vista por versión de desarrollo en la versión de desarrollo anterior el requisito que se ha modificado.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema comprueba que el usuario no tiene permisos para modificar la versión de desarrollo del requisito, lo advierte con un mensaje y acaba el caso de uso.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	Se produce error al grabar y no modifica la versión de desarrollo del requisito, lo advierte con un mensaje y acaba el caso de uso.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.18. GENERAR_DOC_REQUISITO

IDENTIFICACIÓN CASO DE USO				
ID	GENERAR_DOC_REQUISITO			
NOMBRE	Generación automática de un documento de requisitos.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de tipo persona.			
DESCRIPCION	El usuario solicita documento de requisitos y el sistema genera dicho documento de manera automática.			
DISPARADOR	El actor solicita documento de requisito.			
PRECONDICIONES	El requisito ha sido grabado correctamente.			
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	El documento es generado.			
	POSTCONDICION FALLIDA			
	Error en la generación del documento.			
PRIORIDAD				
FRECUENCIA DE USO				
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario solicita generar documento de requisito de un requisito hoja, es decir requisito implementable.		
			2	El sistema consulta si existe plantilla para generar documento.
			3	El sistema genera documento, abriendo herramienta de terceros.
			4	El sistema informa mediante un mensaje al usuario de que se ha generado el documento correctamente y lo muestra tras dar la información.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1a	El usuario solicita generar documento de requisito en un requisito padre de otros requisitos y especifica hasta qué nivel de profundidad quiere obtener el documento.		
			2	El sistema consulta si existe plantilla para generar documento y consulta si el nivel de profundidad seleccionado es posible.

			3	El sistema genera documento, concatenando información de los requisitos hijos, abriendo herramienta de terceros.
			4	El sistema informa mediante un mensaje al usuario de que se ha generado el documento correctamente y lo muestra tras dar la información.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El sistema no tiene plantilla para generar el documento.
			3	El sistema informa al usuario sobre la situación, y la imposibilidad de generar el documento.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema impide al usuario asignar el estado elegido debido al tipo de usuario.
			4	El sistema informa al usuario de la situación y le solicita un nuevo estado
	5	El usuario asigna un nuevo estado y el caso continúa en el punto 3 del camino normal.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			2a	El requisito en cuestión no tiene tanto nivel de profundidad y es necesario avisar al usuario pidiéndole que introduzca un nivel menor.
	3	El usuario introduce un nuevo nivel de profundidad y el caso de uso vuelve al paso 2 del camino alternativo representado por 1a.		
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema da error al abrir herramienta de terceros.
			4	El sistema informa al usuario sobre la situación, y la imposibilidad de generar el documento.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3a	El sistema da error al abrir

				herramienta de terceros.
			4	El sistema informa al usuario sobre la situación, y la imposibilidad de generar el documento.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
			3b	El sistema no puede concatenar datos de los requisitos hijos, porque le faltan datos a requisitos hijos o porque no encuentra información de los mismos.
			4	El sistema informa al usuario sobre la situación, y la imposibilidad de generar el documento.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.19. BÚSQUEDA_REQUISITO

<table>
<tr><th colspan="5">IDENTIFICACIÓN CASO DE USO</th></tr>
<tr><td>ID</td><td colspan="4">BUSQUEDA_REQUISITO</td></tr>
<tr><td>NOMBRE</td><td colspan="4">Búsqueda por criterio como abreviatura, título o descripción.</td></tr>
<tr><th colspan="5">DEFINICIÓN CASO DE USO</th></tr>
<tr><td>ACTOR</td><td colspan="4">Usuario de tipo persona.</td></tr>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario realiza una búsqueda en la que especifica una abreviatura, ó un título, ó una descripción y obtiene una lista de requisitos del sistema.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor accede a la utilidad de búsqueda.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4"></td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">Se muestra al usuario el resultado de su consulta, pudiendo ser vacía, en el caso de que no se encuentren requisitos acordes a los criterios introducidos.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Media</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="8">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El usuario arranca la utilidad de búsqueda y especifica la búsqueda por requisito.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema recoge los datos relativos a los usuarios y los roles sobres los que realizar la consulta y rellena las combos.</td></tr>
<tr><td>3</td><td>El usuario introduce abreviatura y/ó título y/o descripción.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>El sistema valida la existencia del usuario en la base de datos.</td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema realiza la consulta.</td></tr>
<tr><td></td><td></td><td>6</td><td>El sistema muestra por pantalla el resultado de la búsqueda y permite editar el requisito.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>4a</td><td>El usuario especificado no existe.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema muestra el error por pantalla y vuelve a permitir al usuario introducir de nuevo los datos, siguiendo a partir de 3 la ejecución normal.</td></tr>
</table>

EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.20. BUSQUEDA_ROL

IDENTIFICACIÓN CASO DE USO				
ID	BUSQUEDA_ROL			
NOMBRE	Búsqueda por usuario y rol del mismo.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de la aplicación.			
DESCRIPCION	El usuario realiza una búsqueda en la que especifica un usuario y un rol, y el sistema devuelve todos los requisitos en los que ese usuario intervenga con ese rol.			
DISPARADOR	El actor accede a la utilidad de búsqueda.			
PRECONDICIONES				
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	Se muestra al usuario el resultado de su consulta, pudiendo ser vacía, en el caso de que no se encuentren requisitos acordes a los criterios introducidos.			
	POSTCONDICION FALLIDA			
PRIORIDAD	Media			
FRECUENCIA DE USO	Media			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El usuario arranca la utilidad de búsqueda y especifica la búsqueda por rol y usuario.		
			2	El sistema recoge los datos relativos a los usuarios y los roles sobres los que se puede realizar la consulta y rellena las combos.
	3	El usuario introduce el nombre del usuario y el rol por el que filtrar los requisitos en los que esté involucrado. Si se deja en blanco serán todos en los que participe el usuario en		

		cuestión.		
			4	El sistema valida la existencia del usuario en la base de datos.
			5	El sistema realiza la consulta.
			6	El sistema muestra por pantalla el resultado de la búsqueda y permite editar el requisito.
FLUJO ALTERNATIVO				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	4a	El usuario especificado no existe.		
			5	El sistema muestra el error por pantalla y vuelve a permitir al usuario introducir de nuevo los datos, siguiendo a partir de 3 la ejecución normal.
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.21. BUSQUEDA_ROL_ESTADO

IDENTIFICACIÓN CASO DE USO	
ID	BUSQUEDA_ROL_EST
NOMBRE	Búsqueda por rol de usuario y estado.
DEFINICIÓN CASO DE USO	
ACTOR	Usuario de la aplicación.
DESCRIPCION	Búsqueda en la base de datos de requisitos a través del estado del requisito y del rol de usuario.
DISPARADOR	El actor accede a la utilidad de búsqueda.
PRECONDICIONES	
POSTCONDICIONES	POSTCONDICIÓN CORRECTA
	Se muestra al usuario el resultado de su consulta, pudiendo ser vacía, en el caso de que no se encuentren requisitos acordes a los criterios introducidos.
	POSTCONDICION FALLIDA
PRIORIDAD	Media
FRECUENCIA DE USO	Media

<table>
<tr><td rowspan="6">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El actor arranca la utilidad de búsqueda.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema recoge los datos relativos a los usuarios y los roles sobres los que realizar la consulta y rellena las combos.</td></tr>
<tr><td>3</td><td>El usuario introduce el nombre del usuario y el rol por el que filtrar los requisitos en los que esté involucrado especificando que se desea ver solo el trabajo pendiente. Si se deja en blanco serán todos en los que participe el usuario en cuestión.</td><td></td><td></td></tr>
<tr><td></td><td></td><td>4</td><td>El sistema valida la existencia del usuario en la base de datos .</td></tr>
<tr><td></td><td></td><td>5</td><td>El sistema muestra el resultado por pantalla y permite editar el requisito.</td></tr>
<tr><td rowspan="4">FLUJO ALTERNATIVO</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td>2a</td><td>El usuario especificado no existe.</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema muestra el error por pantalla y vuelve a permitir al usuario introducir de nuevo los datos, siguiendo a partir de 3 la ejecución normal.</td></tr>
<tr><td rowspan="4">EXCEPCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>NOTAS</td><td colspan="4"></td></tr>
</table>

3.3.22. CAMBIAR_VISTA_VERSIÓN DE DESARROLLO

IDENTIFICACIÓN CASO DE USO				
ID	CAMBIAR_VISTA_ARQ			
NOMBRE	Cambiar vista a vista por arquitectura.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de la aplicación.			
DESCRIPCION	El usuario selecciona la vista de los requisitos que mejor de adecua a sus necesidades.			
DISPARADOR	El actor accede al menú de vistas.			
PRECONDICIONES				
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	Se actualiza el interfaz de la aplicación.			
	POSTCONDICION FALLIDA			
PRIORIDAD	Baja			
FRECUENCIA DE USO	Media			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El actor selecciona la vista por arquitectura.		
			2	El sistema realiza una nueva consulta sobre la base de datos agrupando el resultado por el path del componente al que esté asociado.
			3	El sistema limpia la vista actual.
			4	El sistema reconstruye el árbol de requisitos. Y lo muestra por pantalla.
	5	El actor obtiene la nueva vista.		
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.23. CAMBIAR_VISTA_JER

IDENTIFICACIÓN CASO DE USO				
ID	CAMBIAR_VISTA_JERARQUICA			
NOMBRE	Cambiar vista a vista por jerarquía.			
DEFINICIÓN CASO DE USO				
ACTOR	Usuario de la aplicación.			
DESCRIPCION	El usuario selecciona la vista de los requisitos organizada por jerarquía.			
DISPARADOR	El actor accede al menú de vistas.			
PRECONDICIONES				
POSTCONDICIONES	POSTCONDICIÓN CORRECTA			
	Se actualiza el interfaz de la aplicación.			
	POSTCONDICION FALLIDA			
PRIORIDAD	Alta			
FRECUENCIA DE USO	Muy Alta			
FLUJO NORMAL DE EJECUCIÓN				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
	1	El actor selecciona la vista por organizada por jerarquía.		
			2	El sistema realiza una nueva consulta sobre la base de datos agrupando el resultado por sus relaciones verticales con padre y posibles hijos.
			3	El sistema limpia la vista actual.
			4	El sistema reconstruye el árbol de requisitos. Y lo muestra por pantalla.
	5	El actor obtiene la nueva vista.		
EXCEPCION				
	ACCIONES ACTOR		RESPUESTAS DEL SISTEMA	
NOTAS				

3.3.24. CAMBIAR_VISTA_ARQ

IDENTIFICACIÓN CASO DE USO	
ID	CAMBIAR_VISTA_ARQ
NOMBRE	Cambiar vista a vista de arquitectura
DEFINICIÓN CASO DE USO	
ACTOR	Usuario de la aplicación.

<table>
<tr><td>DESCRIPCION</td><td colspan="4">El usuario selecciona la vista de los requisitos organizada por módulos.</td></tr>
<tr><td>DISPARADOR</td><td colspan="4">El actor accede al menú de vistas.</td></tr>
<tr><td>PRECONDICIONES</td><td colspan="4"></td></tr>
<tr><td rowspan="4">POSTCONDICIONES</td><td colspan="4">POSTCONDICIÓN CORRECTA</td></tr>
<tr><td colspan="4">Se actualiza el interfaz de la aplicación.</td></tr>
<tr><td colspan="4">POSTCONDICION FALLIDA</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>PRIORIDAD</td><td colspan="4">Media</td></tr>
<tr><td>FRECUENCIA DE USO</td><td colspan="4">Alta</td></tr>
<tr><td rowspan="7">FLUJO NORMAL DE EJECUCIÓN</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td>1</td><td>El actor selecciona la vista por organizada por módulos</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>El sistema realiza una nueva consulta sobre la base de datos agrupando el resultado por sus componentes</td></tr>
<tr><td></td><td></td><td>3</td><td>El sistema limpia la vista actual.</td></tr>
<tr><td></td><td></td><td>4</td><td>El sistema reconstruye el árbol de requisitos. Y lo muestra por pantalla.</td></tr>
<tr><td>5</td><td>El actor obtiene la nueva vista.</td><td></td><td></td></tr>
<tr><td rowspan="4">EXCEPCION</td><td colspan="4"></td></tr>
<tr><td colspan="2">ACCIONES ACTOR</td><td colspan="2">RESPUESTAS DEL SISTEMA</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>NOTAS</td><td colspan="4"></td></tr>
</table>

3.4. ATRIBUTOS DE UN REQUISITO

En este apartado se va a describir la información referente al objeto Requisito. Para organizar la información que se quiere guardar del requisito, qué es necesario y qué no es necesario, se analizan en este punto, junto con los casos de uso, la información que debe de guardarse para efectuar una gestión de los requisitos acorde con los propia ERS y los Casos de Uso de esta fase de Análisis.

No se pretende adelantarse a la fase de Diseño, sino introducir una primera definición de los datos que van asociados al objeto Requisito. En el Diseño se analizarán que clases serán necesarias para manejar la información extraída en este punto.

I. Título del requisito/Abreviatura

La abreviatura se utilizará como un identificar breve de un requisito.

II. Tipo de requisito:

Valores: [TECH, HRMS, BOTH (Collaboration), BOTH (Impact)]

Describe que departamento implementará – o deberá hacer cambios en su desarrollo – a causa del desarrollo de éste requisito. BOTH (Collaboration), por colaboración significaría que implica trabajo tanto de desarrollo tecnológico (TECH) como funcional (HRMS). BOTH (Impact) supone que el requisito, aunque se implementa solo en el lado tecnológico, o en el funcional, obligará al otro a hacer cambios en desarrollos existentes.

III. Descripción:

Descripción del requisito a implementar.

IV. Versión

Versionado del requisito. Del tipo AAA.BBB.CCC.DDD donde:

AAA=Versión mayor de la aplicación donde se implementará el requisito (relación unívoca con una *versión de desarrollo* de código).

BBB=ID del requisito. Los Ids se asignan por orden de alta de los requisitos.

CCC=Revisión mayor. Se cambia manualmente cuando un requisito se revisa en profundidad.

DDD=Revisión menor (automático, el sistema lo incrementa con cada cambio de cualquiera de los atributos del requisito).

Ejemplos:

064.012.003.008 -> Requisito num. 12 de 6.4, versión 3 revisión 8

071.003.001.012 -> Requisito num. 3 de 7.1, versión 1 revisión 12

Todos los números de versión empezarán en 1.

En caso de superar los 999 requisitos, el siguiente requisito sería el 1000, el sistema estará preparado, para modificar la nomenclatura inicial de versión (AAA.BBB.CCC.DDD).

V. Enlaces a documentación:

Valores: [Análisis, Diseño, Codificación, Usuario, Plan de test]

Los documentos relacionados con un requisito (análisis, diseño, prototipos, documentación de usuario…) se guardarán en un directorio físico de un servidor de ficheros. En la base de datos de requisitos se guardará únicamente un enlace a cada uno de ellos.

VI. Relaciones con personas:

Valores: [QA,Desarrollador, Product, Creador, Resp QA, Resp DEV]

Un requisito tendrá una serie de personas que pueden actuar con respecto él. Cada persona tiene un rol. Las comunicaciones de cambios y los permisos sobre determinadas acciones se harán en base a ésta distribución de roles.

VII. Departamento origen:

Valores: [QA, funcional, cliente, preventa]

Esto podría documentarse indirectamente a partir de la persona que crea el requisito.

VIII. Estado:

Valores: [abierto, chequeado, analizado, desechado, codificado, apartado, testeado, cerrado]

Describen el estado de desarrollo actual del requisito.

IX. Comment to status:

Posibilidad de incluir notas que sirvan de comentario a los cambios estado.

X. Relaciones con otros requisitos:

Valores: [Horizontal, Vertical]

Las relaciones verticales modelan la jerarquía de requisitos y las horizontales se refieren a vínculos no jerárquicos entre ellos.

Ejemplos de relaciones horizontales entre requisitos pueden ser:

- La *evolución* o *extensión* de un requisito: Un requisito que amplía la funcionalidad implementada según otro anterior ya desarrollado.

- Relación de *impacto* o *consecuencia*: Un requisito de instalación o configuración, por ejemplo, suele tener como origen otro de funcionalidad.

En general las relaciones jerárquicas permiten ir desde los requisitos de alto nivel a otros más detallados para llegar finalmente a los requisitos de implementación que se usarán como base en el diseño. Las relaciones horizontales se usan para modelar cualquier otro tipo de relación entre requisitos.

XI. Gestión de Cambios:

Valores: [Persona que solicita el cambio]

Cuando el usuario grabe un requisito se guardará en el histórico la información del cambio junto con la persona que lo introdujo, fecha y comentarios. Es posible y recomendable incluir links al foro donde puedan verse las razones que han llevado a aceptar el cambio.

XII. Prioridad inicial:

Valores: [Not Urgent, Urgent,Very Urgent, Critical]

Prioridad asignada por el usuario que da de alta el requisito.

XIII. Prioridades sugeridas:

Valores: [Not Urgent, Urgent,Very Urgent, Critical]

Cuando se plantea si un requisito debe o no ser incluido en la planificación de una nueva versión, se debe establecer la prioridad definitiva del mismo. Para esto se inicia un proceso de negociación en que cada uno de los involucrados en la planificación de la versión va asignando prioridades a cada uno de los requisitos.

XIV. Prioridad development:

Valores : [1-5]

La prioridad final resultante de la negociación del plan de producto queda como prioridad final del requisito durante toda la fase de desarrollo. Normalmente alguien con el rol de product manager es quien establece ésta prioridad definitiva.

XV. Fecha de alta:

Fecha en la que se da de alta el requisito.

XVI. Versión de desarrollo donde se realizará el requisito:

Este campo, además de la estimación, se usará para filtrar requisitos por *versión de desarrollo*. Una *versión de desarrollo* normalmente define una nueva versión de desarrollo del producto.

XVII. Categoría del requisito:

Valores: [N/A, Funcionalidad, Tecnología]

Se refiere al área relacionada con el desarrollo del requisito.

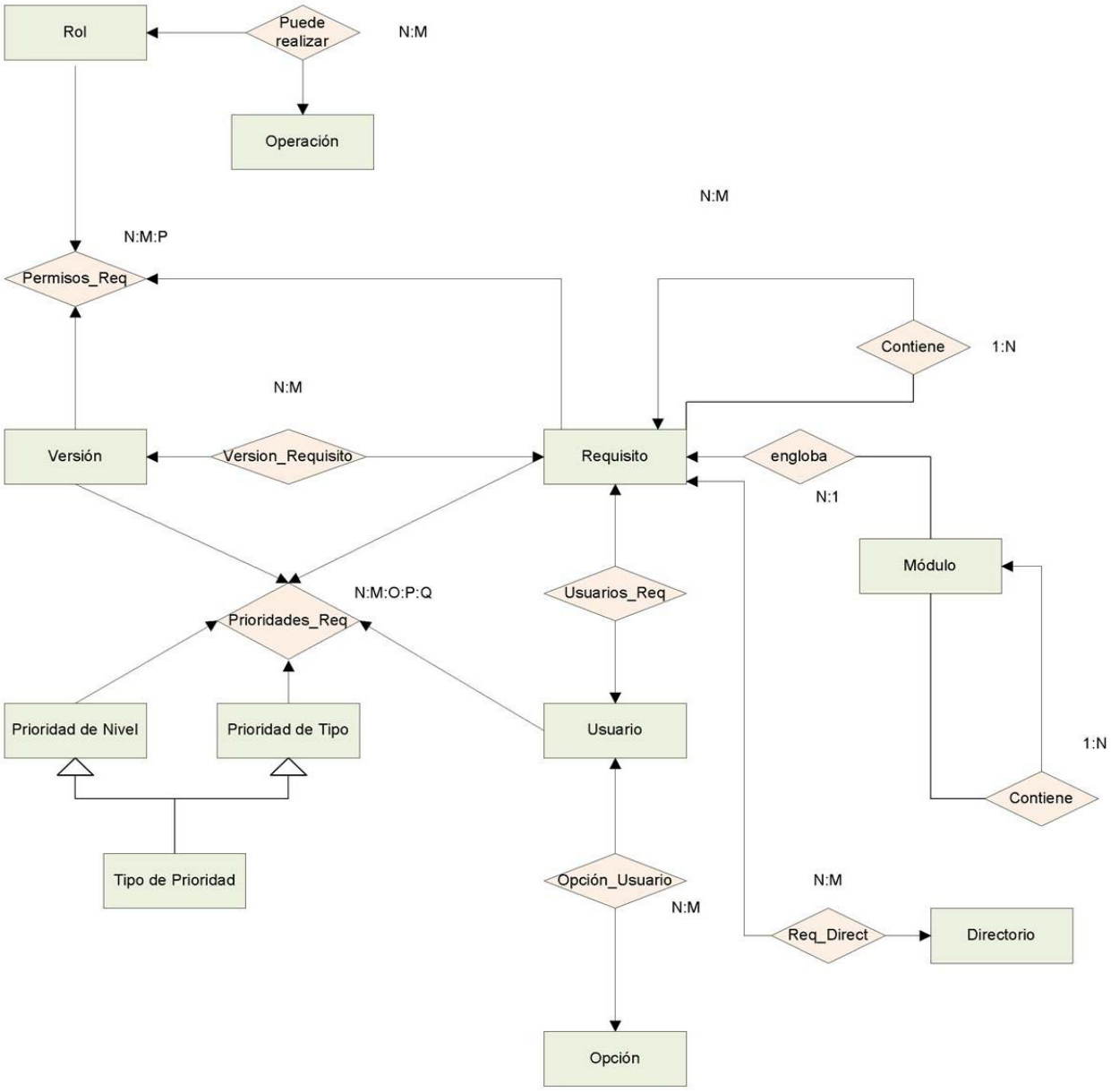
XVIII. Componente de alto nivel:

Componente de arquitectura de alto nivel, módulo, al que pertenece el requisito.

3.5. DIAGRAMAS DE FASE DE ANÁLISIS

3.5.1. Modelo Entidad Relación

El modelo Entidad Relación (E/R), propuesto por Peter P.Chen, y según el autor puede ser usado como una base unificada de los datos. No se puede considerar que exista un modelo E/R único, sino más bien una familia de modelos. En este modelo se muestran las diferentes entidades y sus relaciones, del Sistema de Gestión de Requisitos.



3.5.2. Descripción Entidades Modelo E/R

- Requisito
  - Descripción: Contiene toda la información correspondiente a un requisito
  - Atributos:
    - ID: tipo numérico, PRIMARY KEY
    - Título : tipo cadena
    - Abreviatura: tipo cadena
    - Tipo: tipo enumerado(funcional, tecnológico)
    - Categoría: tipo enumerado (TECH, HRMS, BOTH (Collaboration), BOTH (Impact))
    - Descripción: tipo cadena
    - Versión: tipo cadena
    - Departamento: tipo enumerado (QA, funcional, cliente, preventa)
    - Estado: tipo numérico, FOREIGN KEY
    - Versión de desarrollo: tipo cadena
    - Fecha de Creación
    - Módulo: tipo numérico, FOREIGN KEY
    - Creador: tipo numérico, FOREIGN KEY
- Usuario
  - Descripción: Contiene la información a almacenar sobre los usuarios.
  - Atributos:
    - ID: tipo numérico, PRIMARY KEY
    - Nombre: tipo cadena
    - Login: tipo cadena
    - Password: tipo cadena
    - Departamento: tipo enumerado (QA, funcional, cliente, preventa)
- Opción
  - Descripción: Contiene la información a almacenar sobre las opciones de usuario al acceder al sistema.

- Atributos:
  - ID: tipo numérico, PRIMARY KEY
  - Nombre opción: tipo cadena
  - Descripción: tipo cadena

❖ Tipo de prioridad

- Descripción: Contiene la información a almacenar sobre las distintas prioridades de un requisito.
- Atributos:
  - ID: tipo numérico, PRIMARY KEY
  - Nombre: tipo cadena
  - Descripción: tipo cadena

❖ Prioridad de Nivel

- Descripción: Contiene la información a almacenar sobre las prioridades de nivel.
- Atributos: Heredad atributos de Tipo de Prioridad
  - ID: tipo numérico, PRIMARY KEY
  - Nombre: tipo cadena
  - Descripción: tipo cadena

❖ Prioridad de Tipo

- Descripción: Contiene la información a almacenar sobre las prioridades de tipo.
- Atributos
  - ID: tipo numérico, PRIMARY KEY
  - Nombre: tipo cadena
  - Descripción: tipo cadena

❖ Directorio:

- Descripción: Contiene la información a almacenar sobre los directorios de los documentos asociados a un requisito.

- Atributos
  - ID: tipo numérico, PRIMARY KEY
  - *Path*: tipo cadena

❖ Modulo:

- Descripción: Contiene la información a almacenar sobre los componentes de arquitectura, denominados módulos.
- Atributos
  - ID: tipo numérico, PRIMARY KEY
  - Nombre: tipo cadena
  - Descripción: tipo cadena

❖ Operación

- Descripción: Contiene la información a almacenar sobre las operaciones que puede realizar un usuario dependiendo del rol que ocupe.
- Atributos
  - ID: tipo numérico, PRIMARY KEY
  - Nombre: tipo cadena

❖ Versión

- Descripción: Contiene la información a almacenar sobre las diferentes versiones de un requisito. En referencia a la entidad requisito, en dicha entidad se representa toda la información sobre la el requisito, en esta entidad solamente la información susceptible al cambio.
- Atributos:
  - ID: tipo numérico, PRIMARY KEY
  - Tipo: tipo enumerado(funcional, tecnológico)
  - Categoría: tipo enumerado (TECH, HRMS, BOTH (Collaboration), BOTH (Impact))
  - Descripción: tipo cadena
  - Estado: tipo numérico, FOREIGN KEY
  - Modulo: tipo numérico, FOREIGN KEY
  - Modificador: tipo numérico, FOREIGN KEY

- Rol
    - Descripción: Contiene la información a almacenar sobre los diferentes roles que tienen los usuarios.
    - Atributos:
        - ID: tipo numérico, PRIMARY KEY
        - Nombre: tipo cadena
        - Descripción: tipo cadena

3.5.3. Modelo de Clases Conceptual

Un modelo de clases, es la descripción más clara de la estructura estática de un sistema software. La evolución de este modelo, es el exponente más claro, del progreso de su desarrollo. Este modelo de clases conceptual, está centrado en representar los conceptos de información que manejará el Sistema de Gestión de Requisitos. Se representan clases y responsabilidades.

Descripción del Modelo de Clases Conceptual del Sistema de Gestión de Requisitos:

La clase central del modelo es la clase Requisito. Dicha clase se comunicará, entre otras, con clases controladoras. Se representa una clase Controller, y las clases controladoras nacen por generalización de esta clase. Su responsabilidad es ser un nexo entre la lógica del negocio y el *front end*, que estará formado por formularios. Al mismo tiempo se comunicará con la clase Session, clase que almacenará los datos relativos a la sesión. Esta clase es un *singleton,* El patrón *singleton* provee una única instancia de la clase Session gracias a que:

- La propia clase es responsable de crear una sola instancia.
- Permite el acceso global a dicha instancia en cualquier momento
- El constructor de la clase será privado para que no sea instanciable directamente.

Se utilizará una clase Usuario para obtener y almacenar la información relativa a los usuarios existentes y al propio usuario que accede al sistema. Para la información relativa a las diferentes versiones de los requisitos, es decir, el histórico, se empleará la clase Histórico. Se utilizará también una clase para las operaciones con la base de datos, clase que se relacionará con las clases de la lógica del negocio.

Las responsabilidades concretas de las diferentes clases están incluidas en el Modelo de Clases Conceptual o también denominado Modelo de Clases de Alto Nivel, de la fase de análisis, representado en la página siguiente:

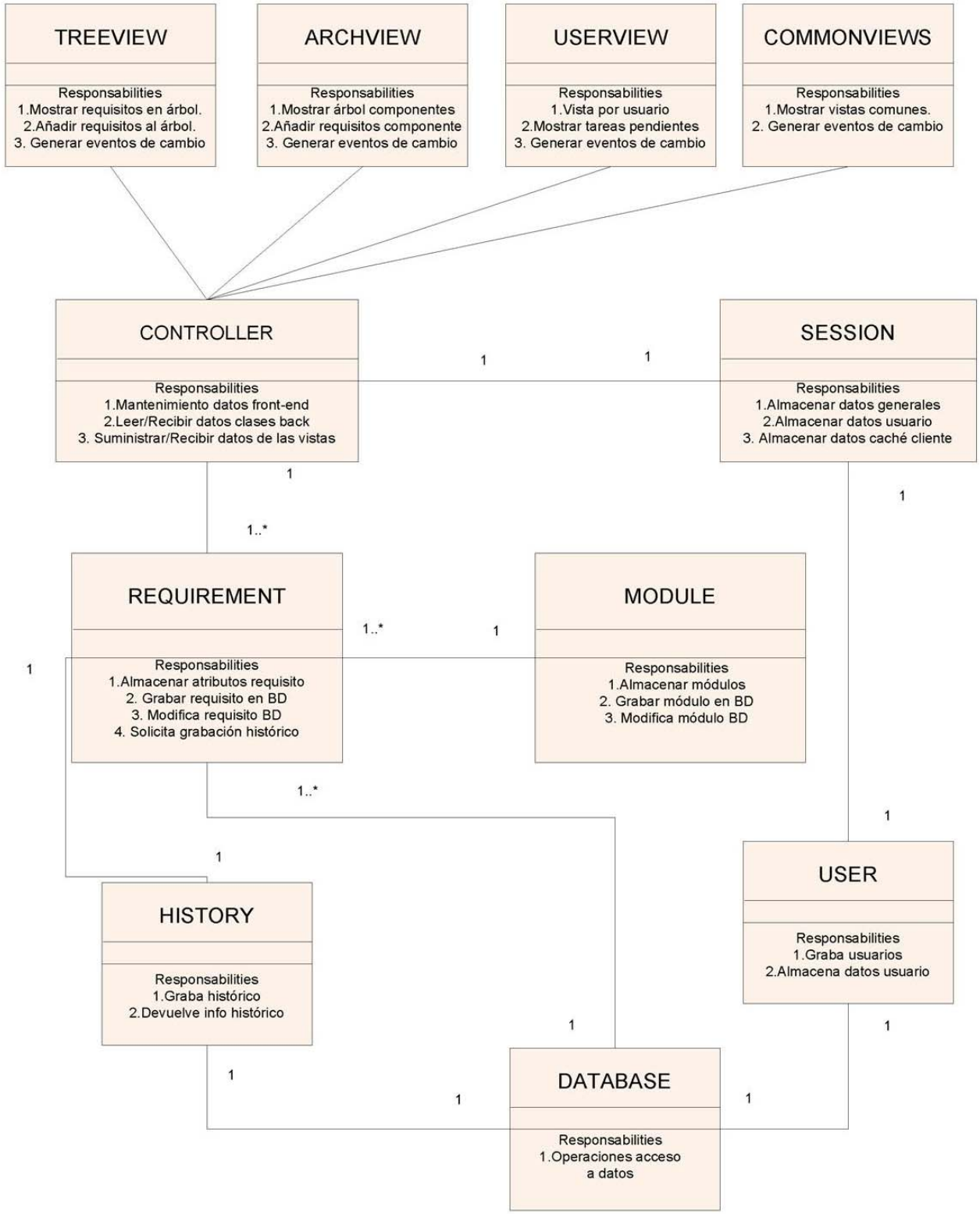
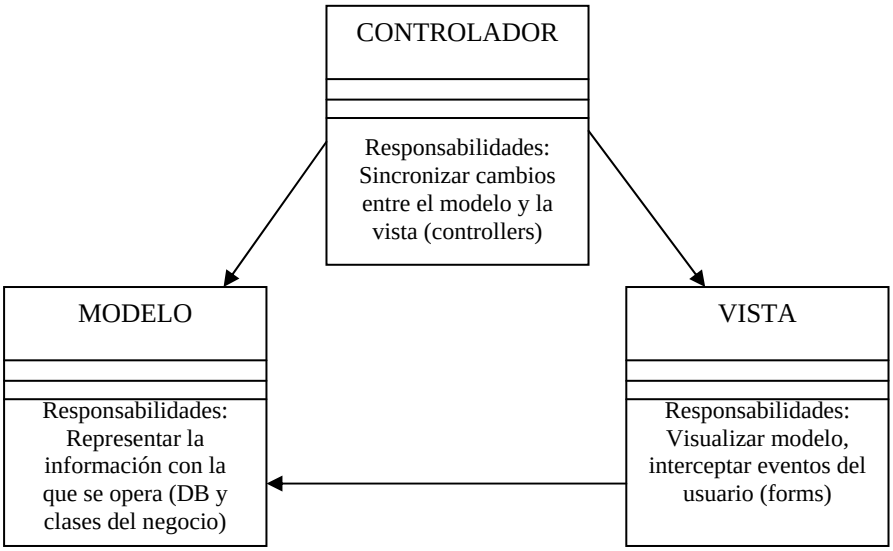


Diagrama de Responsabilidades

3.5.4. Análisis Capa de Presentación. Patrón MVC

La parte cliente del sistema Gestión de Requisitos, se pretende desarrollar bajo el patrón MVC, es decir, Modelo Vista Controlador, el cual tiene por objetivo hacer lo más independientes posibles las capas de Datos junto a la lógica del Negocio, de la lógica de acceso a datos de la capa de presentación.



Modelo: La parte del sistema donde se implementa la lógica de negocio.

Vista: La interfaz con el usuario. Se corresponde con los formularios de presentación que conforman los modos de interacción básicos con el sistema:

- *Vista jerárquica:* La habitual en desarrollo; los requisitos se muestran en un árbol jerárquico, desde los más generales hasta los de implementación.
- *Vista de arquitectura:* En ésta vista la organización atiende a los módulos de sistema a que pertenecen los requisitos.
- *Vistas generales:* Aquí se agrupan formularios auxiliares al sistema: la ventana de acceso, las de mantenimiento de requisitos, las búsquedas, gestión de usuarios del administrador, etc.

Controlador: Mantiene el estado de las vistas y provoca cambios tanto en el Modelo como en la Vista. En principio se intenta conseguir que:

- ✓ El controlador no acceda directamente a controles de los formularios de las vistas, sino que sean los formularios los que reciban datos del controlador y manejen sus propios controles. De ésta forma se disminuye el acoplamiento entre las capas.
- ✓ Que la comunicación con los objetos de negocio en cliente se haga a través del Controlador.

El patrón MVC puede aplicarse a distintas escalas. Se ha aplicado éste patrón a la capa de presentación completa del cliente para buscar un acoplamiento mínimo con los datos y la lógica de negocio.

Justificación del uso del patrón MVC:

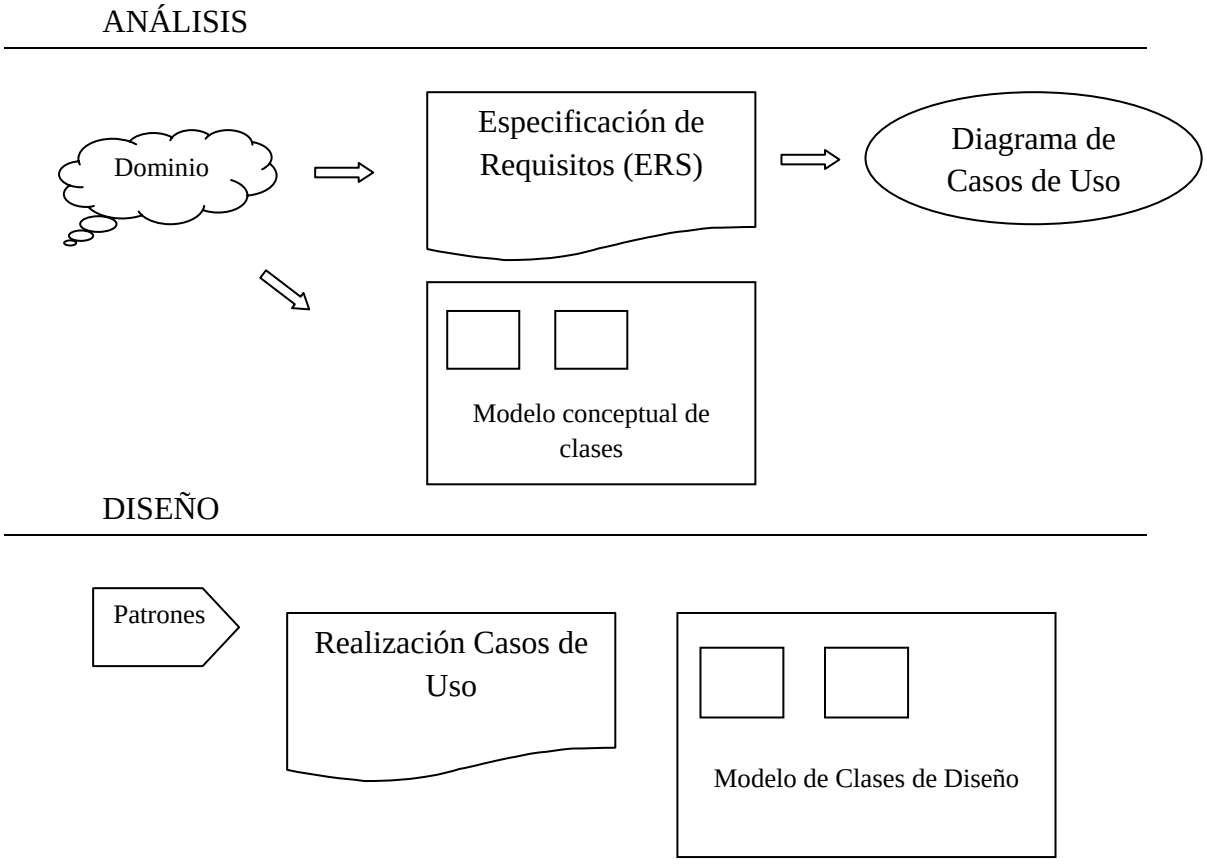
Este planteamiento, siempre deseable viene principalmente impuesto por la oportunidad de desarrollar, en un futuro, una interfaz web para el sistema de gestión de requisitos. Idealmente éste cambio de interfaz afectaría fundamentalmente a las vistas, mientras que las clases de modelo y controladores requerirían muy pocas modificaciones, logrando así que el periodo de desarrollo de la aplicación web fuera menor, al poner en práctica desde un primer momento el patrón MVC.

4. DISEÑO

La fase de Diseño, perteneciente al Modelo del Ciclo de Vida del Software, y desarrollada en cualquier tipo de proyecto de ingeniería, responde a una pregunta diferente respecto al Análisis. Mientras que en la fase de Análisis, se daba respuesta al interrogante ¿Qué?, en esta siguiente fase la pregunta a la que responderá el diseño será ¿Cómo?

En este apartado se muestran las diferentes técnicas y diagramas para el modelado de Diseño, siempre apoyadas en la notación UML.

A continuación, a modo introductorio se muestra la relación entre diagramas UML para las dos primeras fases del ciclo de vida:



El Sistema de Gestión de Requisitos se ha subdividido en subsistemas, debido al amplio tamaño del mismo, con el objetivo de lograr un diseño más directo y detallado de cada parte de las que se compone:

Se han identificado 5 subsistemas:

I. Subsistema de Acceso a Datos

Clases que implementan la capa de acceso a datos, conocida también como DAL (Data Access Layer).

II. Subsistema de Lógica de Negocio en Servidor

Clases pertenecientes a la parte servidora del sistema. Las clases requisito y módulo tienen su réplica en este subsistema, junto con las clases encargadas de la seguridad y usuarios.

III. Subsistema Lógica de Negocio en cliente

Clases pertenecientes a la parte cliente del sistema. Se relacionan con la parte de presentación por medio de los controladores. Gestión de sesión.

IV. Subsistema de Presentación

Clases pertenecientes a la parte cliente. Interfaz del usuario, bajo patrón MVC.

En siguientes apartados se muestra el diseño de cada uno de los subsistemas definidos anteriormente, dónde se muestran las consideraciones tenidas en cuenta relativas al diseño, y la representación de las clases que forman cada uno de los subsistemas por medio de Modelos de Clases de Diseño, según el lenguaje unificado de modelado (UML)

4.1. REALIZACIÓN DE CASOS DE USO

La asignación de responsabilidades y el diseño de colaboraciones son etapas muy importantes y creativas durante el diseño, mientras se elaboran los diagramas, o mientras se programa.

La realización de un caso de uso describe cómo se realiza un caso de uso en particular en el modelo de diseño, en función de los objetos que colabora.

CASO DE USO		CLASES RESPONSABLES	
		PRINCIPAL	AUXILIARES
3.1	**LOGIN_APLICACION**: Login en la aplicación	Session	
3.2	**ALTA_REQUISITO**: Alta de un requisito.	Requirement	Session, Module, History
3.3	**EDICIÓN_REQUISITO**: Edición y modificación de un requisito.	Requirement	Session, Module, History
3.4	**MODIFICACIÓN_ATRIBUTOS**: Modificación de atributos base del requisito.	Requirement	Session, Module, History
3.5	**ASOCIAR_DOCUMENTO**: Asociar documento a un requisito.	Requirement	Session, Module, History
3.6	**MODIFICACIÓN_ESTADO**: Modificar estado del requisito.	Requirement	Session, Module, History
3.7	**MODIFICACIÓN_ESTADO_REQ_SUBREQ**: Modificar estado de algunos hijos no modifica padre.	Requirement	Session, History
3.8	**MODIFICACIÓN_ESTADO_REQ_LASTSUBREQ**: Modificar estado de último sub-requisito hijo cambia padre.	Requirement	Session, User, History
3.9	**MODIFICACIÓN_ESTADO_REQ_PADRE**: Modificar estado de requisito padre, modifica los hijos.	Requirement	Session, User, History
3.10	**MOVER_REQUISITO**: Cambiar requisito padre de un requisito.	Requirement	Module
3.11	**MOVER_REQUISITO_COMP**: Mover requisito de componente, desde la vista de arquitectura de componentes o desde la combo estando en la vista de requisitos.	Requirement	Session, User, History
3.12	**RELACIONAR_REQUISITO**: Relacionar requisitos con otros requisitos.	Requirement	Session, History
3.13	**SUGERIR_PRIORIDAD**: Sugerir prioridad de implementación	Requirement	Session, User, History
3.14	**CAMBIAR_PRIOR_SUGERIDA**: Cambiar la prioridad de implementación sugerida	Requirement	Session, User, History

3.15	**FIJAR_PRIORIDAD_DEFINITIVA**: Fijar la prioridad de implementación.	Requirement	Session, User, History
3.16	**ASIGNAR_PARTICIPANTES**: Asignar participantes al requisito.	Requirement	Session, User
3.17	**MODIFICAR_VERSIÓN DE DESARROLLO**: Modificación de la versión de desarrollo a la que está asignado un requisito.	Requirement	Session, User, History
3.18	**GENERAR_DOC_REQUISITO**: Generación automática de un documento de requisitos.	Requirement	Report
3.19	**BUSQUEDA_REQUISITO**: Buscar requisito	Requirement	User
3.20	**BUSQUEDA_ROL**: Búsqueda por rol.	Requirement	User
3.21	**BUSQUEDA_ROL_EST**: Búsqueda por rol y estado.	Requirement	User
3.22	**CAMBIAR_VISTA_VERSIÓN DE DESARROLLO**: Cambiar vista a vista por versión de desarrollo.	CommonViews	Requirement
3.23	**CAMBIAR_VISTA_ARQ**: Cambiar vista a vista por arquitectura.	CommonViews	Requirement
3.24	**CAMBIAR_VISTA_JER**: Cambiar vista a vista por jerarquía.	CommonViews	Requirement

4.2. ARQUITECTURA DEL SISTEMA

4.2.1. Aspectos de Arquitectura

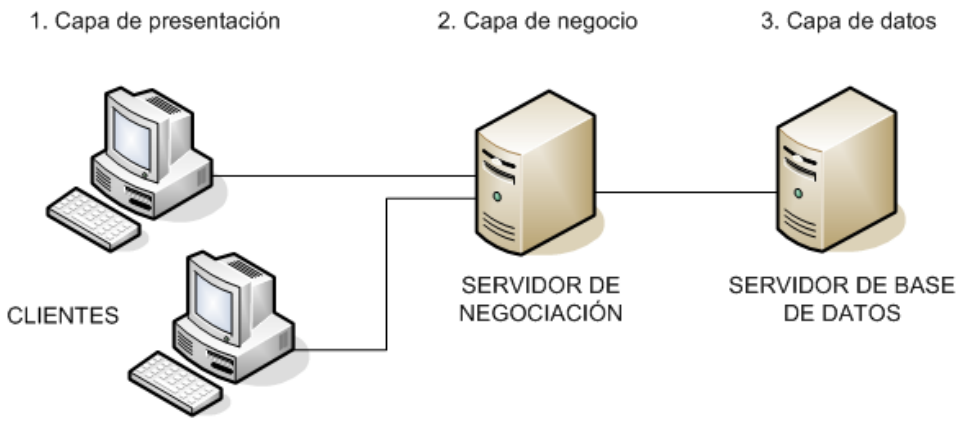
- Topología tres capas

  La principal ventaja de utilizar una programación por capas radica, en la posibilidad de que si se deben de realizar en un futuro cambios en la aplicación, se ataca directamente al nivel o capa afectado. Cada nivel posee una misión simple, lo que permite el diseño de arquitecturas escalables.

  - ✓ **Capa de presentación:** Interfaz que ve el usuario. Presenta el sistema al usuario, le comunica la información y captura la información del usuario.
  - ✓ **Capa de negocio:** Se denomina también capa de lógica del negocio, porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
  - ✓ **Capa de datos:** Gestor de base de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Aquí residen los datos.

  Todas estas capas pueden residir en un único ordenador. Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

  Teniendo en cuenta la posibilidad de que el Sistema Gestión de Requisitos, en una nueva iteración, pudiera sufrir cambios, se decide optar por un desarrollo en 3 capas.



Programación en 3 capas

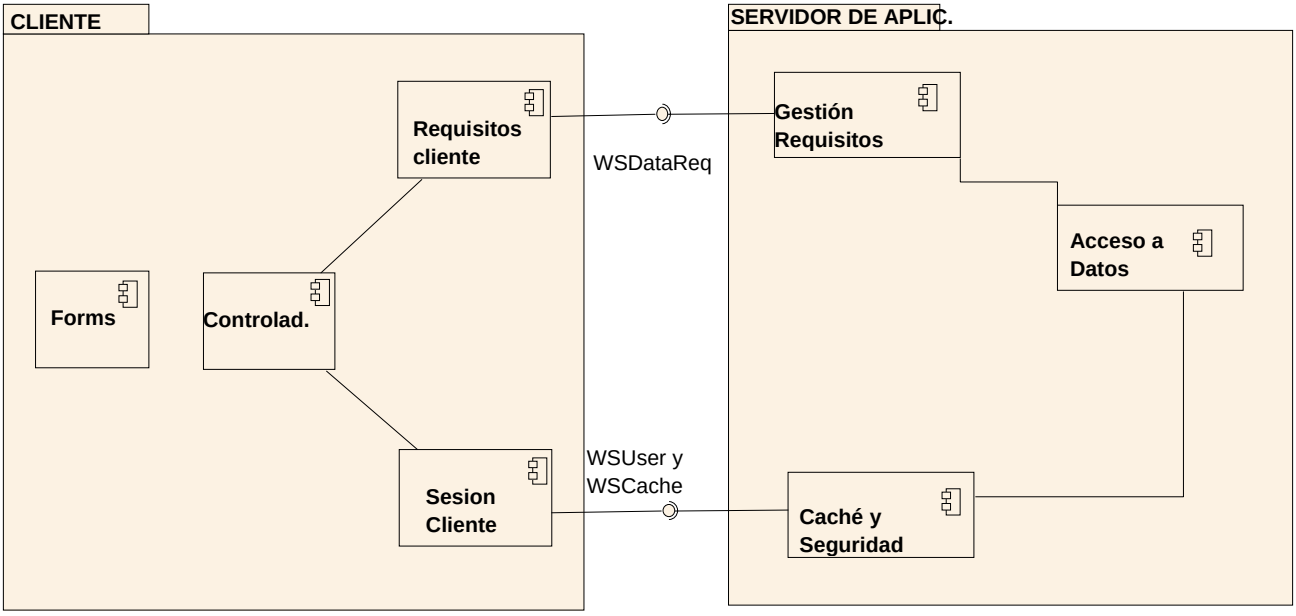
- Uso de servicios web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una de las opciones es hablar de los Servicios Web como un conjunto de aplicaciones o de métodos con capacidad para inter-operar en la Web. Estas aplicaciones o métodos intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

La aplicación cliente del Sistema de Gestión de Requisitos se comunica con la parte servidora que se ejecuta en un servidor web exponiendo al cliente una serie de servicios web que permiten acceder a la funcionalidad de negocio. Esta comunicación se realiza a través de Servicios Web.

En todo el proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (Protocolo Simple de Acceso a Objetos). Protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP , SMTP , etc. SOAP especifica el formato de los mensajes.

4.2.2. Diagrama de Arquitectura



En la aplicación cliente, ubicamos el Subsistema de Presentación que contendrá las clases formularios, y las clases controladoras, el Subsistema de Lógica de Negocio en Cliente. En la parte servidora (servidor de aplicaciones) localizaremos el Subsistema de Acceso a Datos, dónde se incluirá la clase Database, y el Subsistema de Lógica de Negocio en Servidor, dónde estarán replicadas las clases Requisito y Módulo de la aplicación cliente.

4.3. DIGRAMA DE ESTADOS.CICLO DE VIDA DEL REQUISITO

En este diagrama de estados, se representan los diferentes estados por los que atraviesa un requisito, desde que se crea hasta que se decide implementar o denegar. Partiendo de la definición de los atributos del requisito, los diferentes estados del ciclo de vida de un requisito son:

- ***Abierto*:**

  Uno de los dos estados iniciales posibles. Es el estado más temprano dentro del ciclo de vida de un requisito. El requisito, no ha sido aceptado aún para continuar con su desarrollo. Puede ser desechado, o aceptado.

- ***Chequeado*:**

  Segundo estado inicial posible de un requisito. El requisito es aprobado para poder continuar su ciclo de vida.

- ***Analizado*:**

  Estado de análisis. El requisito es analizado, para decidir sobre si se desarrolla o no se desarrolla.

- ***Desechado*:**

  Estado de denegación de requisito. Implica que el requisito no continúa con su ciclo de vida.

- ***Codificado*:**

  Estado que indica que el requisito pasa a implementarse, se ha decidido afirmativamente sobre su desarrollo.

- ***Apartado*:**

  Estado que describe que el requisito no se decide, por el momento, implementarse. Queda en este estado hasta que se decida implementar, o desechar.

- ***Testeado*:**

  El requisito está en estado de prueba. Se pone a prueba si ha sido implementado y es eficaz**.**

- ***Cerrado*:**

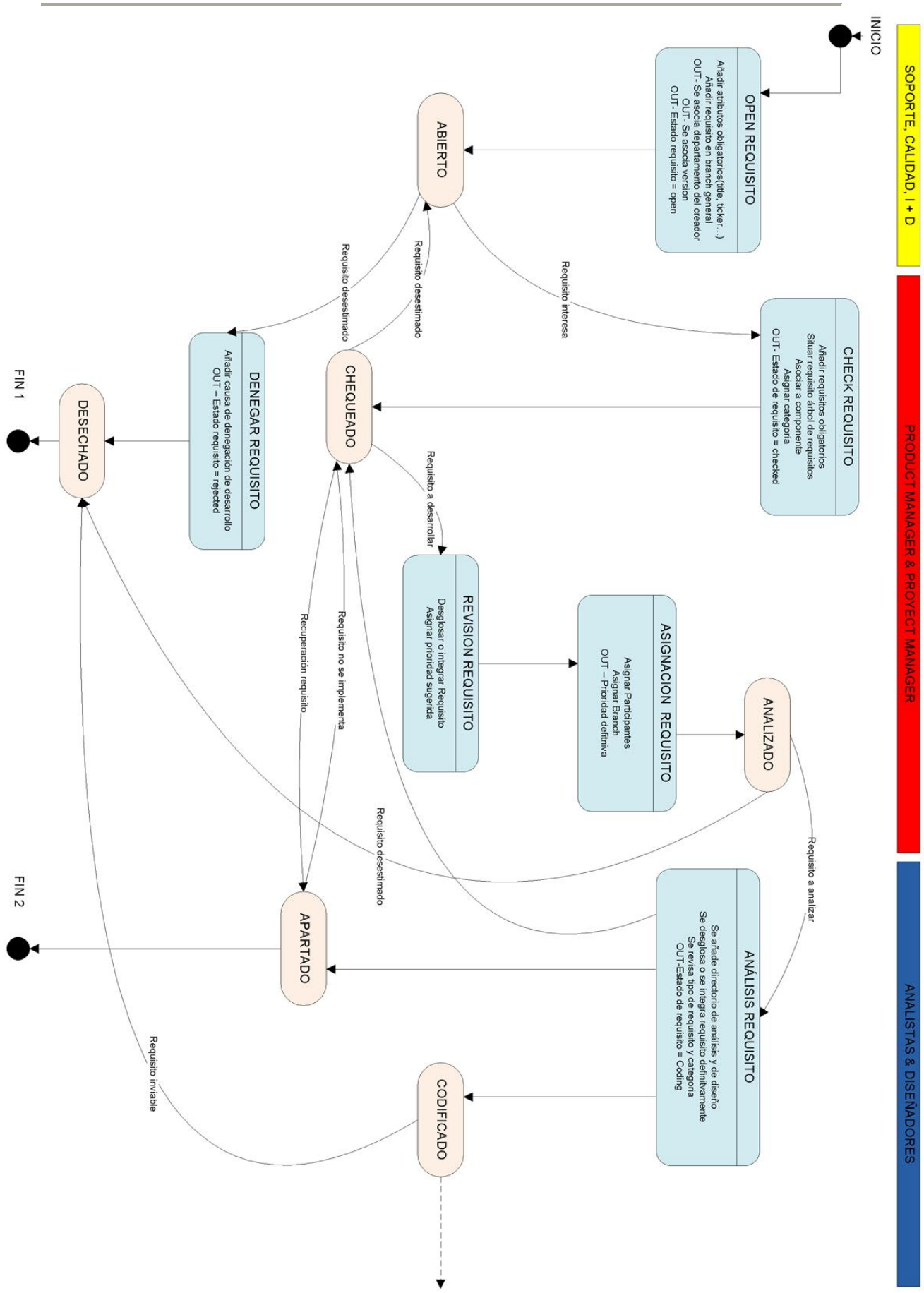
  Estado final del ciclo de vida de un requisito. El estado fue probado tras implementarse, y tras las pruebas pasa a este estado. Recordar que un requisito siempre permanecerá en el sistema, y este es su estado final “eterno”.

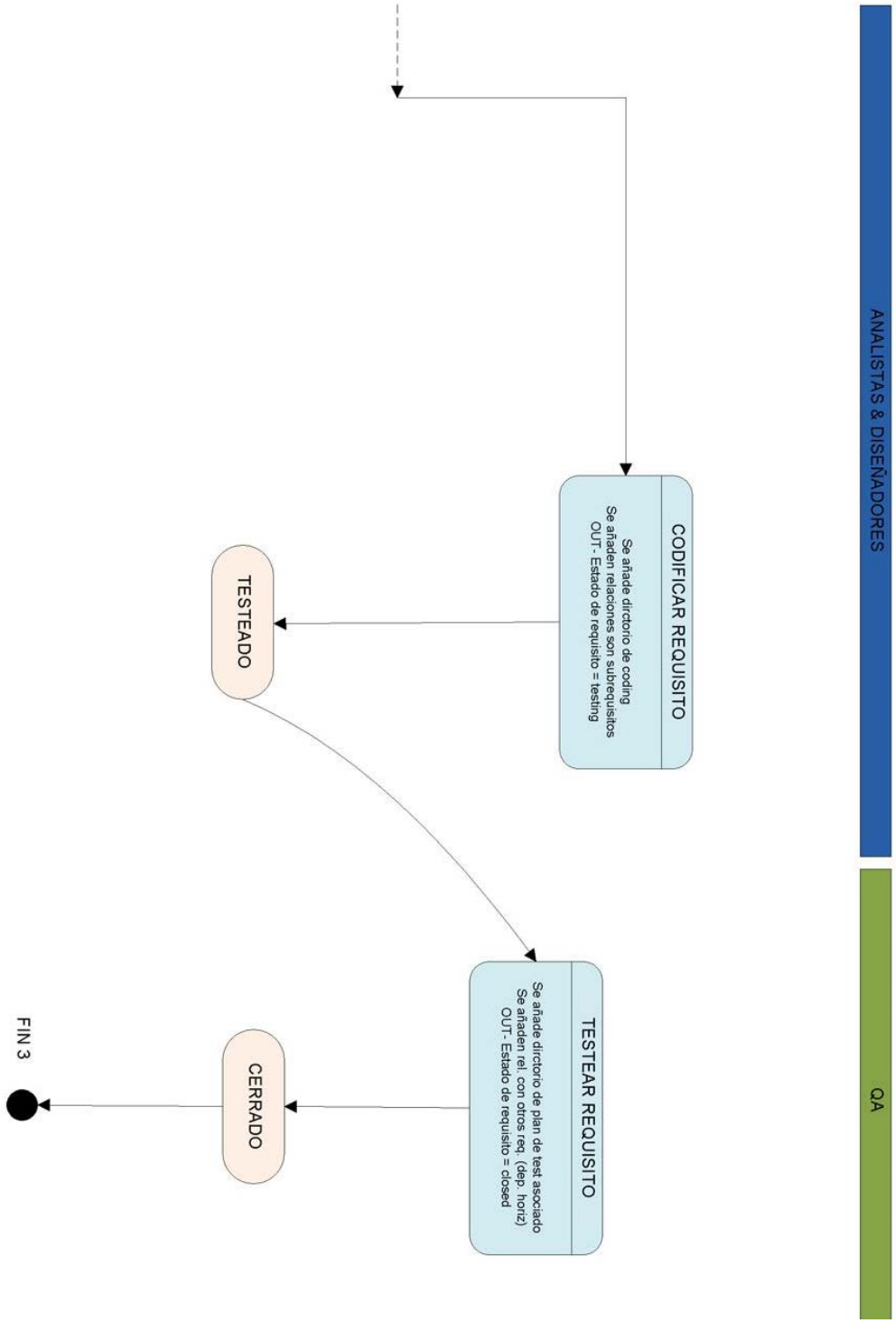
Cada uno de ellos define el estado de desarrollo de un requisito introducido en el sistema. Recordar que un requisito introducido en el sistema, nunca será eliminado, como estado final se quedará en estado CERRADO.

Descripción del ciclo de vida del requisito

Cuándo un requisito se crea en el sistema, el requisito nace con el estado ABIERTO. Si el requisito interesa, decisión que tomará un Project Manager, el requisito pasará a estado CHEQUEADO, y estará asociado a un componente de arquitectura (a un módulo). Si el requisito no interesa, será denegado y pasará a estado DESECHADO. El requisito se revisará, se le asignarán participantes y se valorará si se debe desarrollar. Si no se decide desarrollar pasará de nuevo a ABIERTO, y en caso de que sí se decida desarrollar, evaluando las diferentes prioridades que marcaron los componentes del *team* (participantes), el requisito pasa a estado ANALIZADO. El requisito entrará en fase de Análisis y Diseño y se agregarán los respectivos documentos respectivos de cada fase. Si el requisito se decide no implementar, el requisito pasará a estado APARTADO, y se acabará el ciclo de vida. Si por el contrario se decide implementar, el requisito adquiere el estado de CODIFICADO. En este estado, se agregará al finalizar la implementación, el documento de CODIFICADO, y se establecerán si se cree necesario, relaciones con sub-requisitos. Tras la implementación, el requisito pasará al estado TESTEADO, y tras la realización de los casos de test, se agregará el documento correspondiente de TESTEADO. Se podrán establecer relaciones horizontales con otros requisitos. Tras realizar correctamente la fase de pruebas, el requisito pasará a estado CERRADO y permanecerá en ese estado perpetuamente.

En las dos páginas siguientes se muestra el diagrama de estados, dividido en dos partes para su correcta legibilidad.


SOPORTE, CALIDAD, I + D
PRODUCT MANAGER & PROYECT MANAGER
ANALISTAS & DISEÑADORES
INICIO
OPEN REQUISITO
Añadir atributos obligatorios(title, ticker..)
Añadir requisito en branch general
OUT- Se asocia departamento del creador
OUT- Se asocia version
OUT- Estado requisito = open
ABIERTO
CHECK REQUISITO
Añadir requisitos obligatorios
Situar requisito árbol de requisitos
Asociar a componente
Asignar categoría
OUT - Estado de requisito = checked
Requisito interesa
Requisito desestimado
Requisito desestimado
CHEQUEADO
DENEGAR REQUISITO
Añadir causa de denegación de desarrollo
OUT – Estado requisito = rejected
DESECHADO
FIN 1
Requisito a desarrollar
REVISION REQUISITO
Desglosar o integrar Requisito
Asignar prioridad sugerida
ASIGNACION REQUISITO
Asignar Participantes
Asignar Branch
OUT – Prioridad definitiva
ANALIZADO
Requisito a analizar
ANÁLISIS REQUISITO
Se añade directorio de análisis y de diseño
Se desglosa o se integra requisito definitivamente
Se revisa tipo de requisito y categoría
OUT-Estado de requisito = Coding
Recuperación requisito
Requisito no se implementa
Requisito desestimado
APARTADO
FIN 2
Requisito inviable
CODIFICADO


ANALISTAS & DISEÑADORES
QA
CODIFICAR REQUISITO
Se añade directorio de coding
Se añaden relaciones son subrequisitos
OUT- Estado de requisito = testing
TESTEADO
TESTEAR REQUISITO
Se añade directorio de plan de test asociado
Se añaden rel. con otros req. (dep. horiz)
OUT- Estado de requisito = closed
CERRADO
FIN 3

4.4. SUBSISTEMA DE ACCESO A DATOS

4.4.1. Descripción

En la capa de acceso a datos se busca encapsular toda la lógica de aplicación dedicada a recuperar y actualizar información de la base de datos. El objetivo es implementar métodos de acceso a los datos, que nos simplifiquen dicha labor de acceso a la base de datos (métodos de inserción en la base de datos, actualización, consultas etc). La capa de acceso a datos está ubicada en el servidor de aplicaciones y se comunicará con el cliente usando objetos genéricos de datos. Debe gestionar de forma lo más transparente posible tanto la reutilización de conexiones como las transacciones.

El subsistema de acceso a datos, es el único desarrollo del Sistema de Gestión de Requisitos dependiente del Gestor de Base de Datos utilizado. En caso de migración a otro gestor de base de datos, sería necesario sustituir este subsistema, por un subsistema acorde al gestor utilizado. Este subsistema se implementará teniendo como elección de gestor de base de datos Sql Server 2005.

Por otro lado, además de proveer la funcionalidad básica de acceso y modificación de datos, desde éste componente se suministrará al resto de componentes de negocio un repositorio con las tablas que pueden ser modificadas o accedidas desde la aplicación.

Se ha elegido implementar la DAL (Data Access Layer), debido a que las primeras pruebas realizadas sobre accesos a la base de datos, dieron como resultado que, los tiempos de respuesta fueron menores lanzando las sentencias vía ADO.Net, que mediante la utilizando de procedimientos almacenados en Transact SQL. Y por otra parte, el hecho de que si se desea cambiar el gestor de base de datos en un futuro, supone más esfuerzo transformar los procedimientos almacenados en Transact SQL a PL Sql, que adaptar la DAL al nuevo gestor de base de datos que se utilice.

4.4.2. Componentes básicos

- Clase base de Acceso a Datos:
  - Clase que permitirá encapsular la conexión a la base de datos de cada cliente. Desde esta clase se ejecutarán las sentencias genéricas que afecten a varias tablas de la base de datos.
- Tabla Diccionario:
  - Catálogo de tablas de la base de datos. Se guardarán parámetros relativos a la tabla, por ejemplo si es *cacheable* o no.
- Clase tabla:
  - Una clase que encapsula una tabla de la base de datos y suministra la funcionalidad básica de acceso.
- Clase gestora de caché:
  - Clase de la parte servidora, que gestione la caché de tablas y su interfaz como WebService en la parte cliente.
  - En la inicialización del WebService se traerán de la base de datos todas las tablas *cacheables*. La información sobre qué tablas del modelo son o no *cacheables* se guardaría en la tabla Diccionario de la base de datos. Cada cliente tendría acceso a las tablas cacheadas, tras el login.
- Clase tablas de cache:
  - Suministraría un método de sincronización de cachés del servidor, que podría ser llamado en cualquier momento para recibir cambios de datos, sin necesidad de reiniciar el servicio web.

4.4.3. Detalles de implementación

- ✓ Los clientes del subsistema de Acceso a Datos, serán objetos también de la parte servidora, puesto que la DAL estará implementada en la parte servidora del Sistema Gestión de Requisitos. Por si fuera necesario enviar peticiones también desde la parte cliente, se mantiene una interfaz

de acceso también desde dicha parte, aunque el funcionamiento normal será accediendo a este subsistema desde objetos de la propia parte servidora.

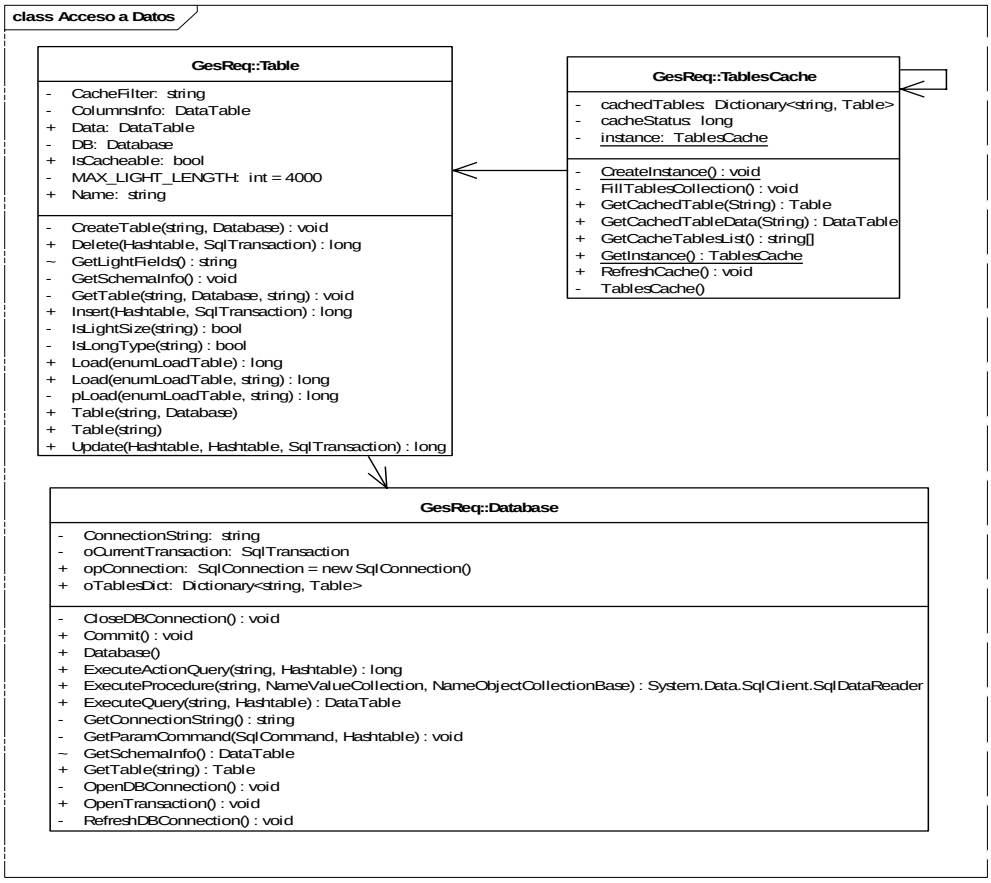
- ✓ Para comunicarse con el cliente, la DAL deberá utilizar objetos de datos *serializables* debido a que el acceso será mediante Servicios Web, debido a esto se utilizan los objetos DataTable como base de la comunicación del resultado de las consultas de acceso a la base de datos entre cliente y los propios servicios.

4.4.4. Modelo de clases del subsistema

La clase principal del subsistema es la clase DataBase. Se ofrece funcionalidad para operar con la base de datos, con métodos de apertura, cierre etc. La clase Table, que se relaciona con la clase Database, incluye los métodos necesarios para operar con tablas de la base de datos.

Por otra parte, se incluye una clase para el almacenaje en caché, de las tablas de la base de datos que pueda interesar que se carguen en caché, cuando se inicie la aplicación en el cliente.

El modelo de representa en la página siguiente, para una mejor legibilidad del mismo.



4.4.5. Métodos principales

Clase: Database, **Método**: OpenDBConnection(), **Desc**: Apertura base de datos

Clase: Database, **Método**: CloseDBConnection(), **Desc**: Cierre base de datos

Clase: Table, **Método**: CreateTable(), **Desc**: Creación tabla

Clase: Table, **Método**: GetTable(), **Desc**: Consulta tabla

Clase: TablesCache, **Método**: GetCachedTable(), **Desc**: Consulta tablas de caché

4.5. SUBSISTEMA DE LOGICA DE NEGOCIO EN CLIENTE

4.5.1. Descripción

El subsistema Lógica de Negocio en cliente estará formado entre otras clases por la clase sesión y por una clase que trate los datos del usuario en el cliente. La clase sesión se utilizará, como su nombre bien indica para almacenar los datos relativos a la sesión. La clase de sesión es una clase que se creará al acceder a la aplicación, y estará accesible en todo momento desde el cliente durante la ejecución del sistema.

Por otra parte, este subsistema contendrá dos clases fundamentales, como son la clase Requisito y la clase Modulo, clases que serán el eje de la lógica del negocio. Las dos clases tienen un comportamiento similar en cuanto a su integración con el sistema caché. Cualquier módulo o requisito solicitado desde cliente se busca primero en caché y, si no se encuentra allí, se carga previamente a enviarlo en respuesta.

4.5.2. Componentes básicos

- Clase sesión:
  - Contendrá la caché de tablas utilizadas en cliente, enviadas desde el servidor cuando se realice el login.
  - Contendrá la cache con los requisitos que se vayan cargando en el cliente. Una vez que se lea un requisito de la base de datos, se mantendrá en la caché y se recuperará desde la misma mientras exista la sesión.
  - Contiene la caché de módulos similar a la de los requisitos.
- Clase usuario cliente:
  - Clase que almacena la información propia de cada usuario, al efectuar login en la aplicación, junto con la información de seguridad relativa al propio usuario (rol, permisos etc.)

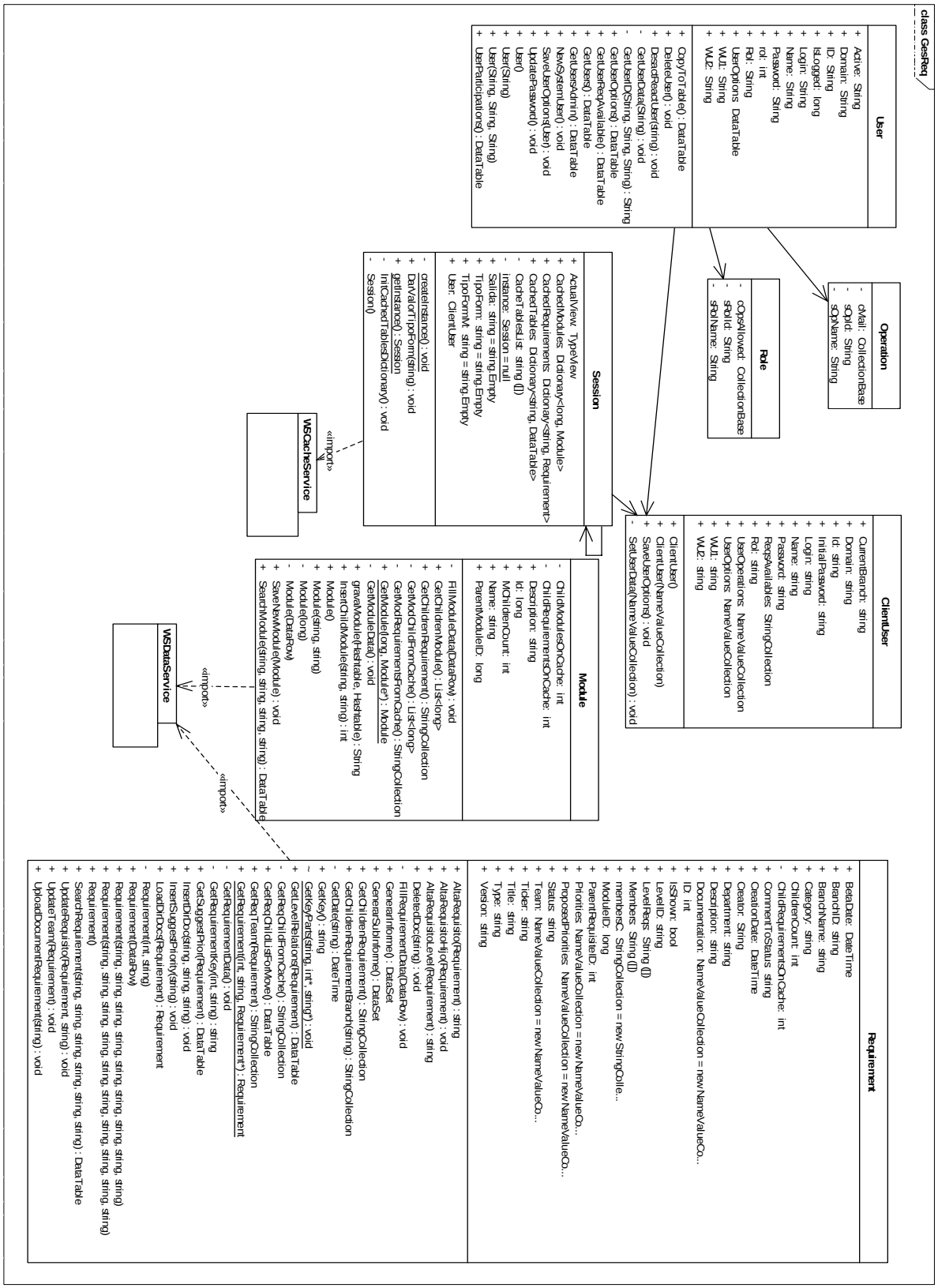
- Clase Requisito:
    - Atributos similares a los que tiene la clase Requisito en la parte servidora, pero diferentes operaciones. Las operaciones consisten básicamente en la solicitud hacia la parte servidora, mediante el servicio Web (Data Service), de datos de los requisitos.
- Clase Modulo:
    - Funcionamiento similar a la clase Requisito, con la peculiaridad de que podrá hacer solicitudes de objetos requisito, debido a que un módulo puede contener otros módulos, pero aglutina requisitos.

4.5.3. Detalles de implementación

- ✓ Con respecto a la aplicación de seguridad en cliente, se intentará en la medida de lo posible reducir al mínimo las transacciones entre el cliente y el servidor. El cliente debería disponer en la carga de los formularios, que operaciones puede realizar el usuario que ha accedido al sistema, en función de sus rol, y qué datos no se deben mostrar en los formularios dependiendo del propio rol. La idea es aplicar la seguridad de manera que se impida que el usuario realice las operaciones que no puede utilizar o los datos que no debe manejar y se evita la situación en la que el servidor deniega una operación por falta de permisos.
- ✓ La clase sesión, es una clase definida de tal manera que pueda ser accedida desde cualquier otra clase del cliente, en cualquier momento de la ejecución, y contiene a la vez un puntero a la clase usuario cliente, es decir, para acceder a los datos del usuario logado, se accede a la clase sesión y posteriormente a la clase usuario cliente.

4.5.4. Modelo de clases del subsistema

La principal clase del modelo de clases del subsistema de lógica de negocio en cliente es la clase Requirement. Es la clase fundamental, para almacenar información de los requisitos, y ofrece métodos para modificar, crear requisitos entre otras funcionalidades establecidas. Además a través de esta clase, el subsistema se relaciona con otros subsistemas, como es el caso del subsistema de Presentación, y con el Subsistema de Lógica de Negocio en Servidor, que se comunicará a través de servicios web. Se destaca también en la lógica de negocio de cliente, la clase Session, clase básica para almacenar la información relativa a la sesión. Entre otra información necesaria a almacenar, se destaca la información relativa al usuario que accede a la aplicación desde un cliente, y para ello se define la clase Client User. Las clases Requirement, Session, Modulo junto con la clase Usuario, acceden a las interfaces de los diferentes servicios web, y se comunican gracias a esto, como se ha citado antes, con la parte servidora del sistema.


class GestReq
User
Operation
Role
ClientUser
Session
Module
Requirement
WSCacheService
WSDataService
«import»
«import»
«import»

4.6. SUBSISTEMA DE LOGICA DE NEGOCIO EN SERVIDOR

4.6.1. Descripción

En el subsistema que contiene Lógica de Negocio en la parte servidora, se agrupa toda la lógica necesaria para trabajar con requisitos y módulos de arquitectura.

Repartir la lógica de negocio entre el cliente y el servidor, implica tener las clases del cliente replicadas en servidor, para poder enviar los datos resultado de las consultas, a través del Servicio Web. Es decir habrá una clase en servidor para tratar los datos de requisitos, y otra para tratar los datos de módulos. Las funciones de carga de datos están diseñadas para responder a las necesidades de la aplicación cliente respecto a requisitos y módulos, optimizando el movimiento de datos a través del servicio web.

En el subsistema de lógica de negocio en servidor, se incluye también gestión de seguridad y usuarios. Para ellos se tendrán una clase que permita la lectura de los datos de los usuarios, y por otra parte una clase que permita la gestión de los mismos (ejemplo: dar de alta usuarios por parte del administrador)

4.6.2. Componentes básicos

- Clase Requisito en Servidor:
  - Tendrá las mismas propiedades prácticamente que la clase Requisito en cliente, pero se utilizarán poco debido a lo comentado anteriormente. De hecho el constructor de la clase sólo estará formado del identificador y la *versión de desarrollo* de desarrollo.
- Clase Módulo en Servidor:
  - El módulo es la unidad básica de arquitectura. Cada módulo tendrá varios requisitos asignados. Se pretenden mostrar estos requisitos, de acuerdo al componente (módulo) al que afectan.
  - La gestión de los módulos compartirá puntos comunes a la de los requisitos, pero será más simplista, debido a que la cantidad información a tratar es menor.

- Un módulo podrá contener otros módulos y/o requisitos asociados como hijos suyos. A diferencia de los requisitos, los módulos no están organizados por versión de desarrollo, eso se debe a que ofrecen una visión estática de la aplicación.

- Clase Usuario:
  - Clase que se implementará para poder leer todos los usuarios al ejecutar la aplicación y arrancar el servicio Web.
  - Se utiliza también para recuperar las opciones de usuario, una vez validado el mismo en el login.

- Clase Gestor de Usuarios:
  - Permite entre otras cosas la posibilidad de dar de alta usuarios nuevos por partes del administrador.
  - Gestiona las operaciones de acceso a los datos de usuarios.

4.6.3. Detalles Implementación

- ✓ El ciclo de vida del requisito en servidor es muy corto. Las clases de Requisito y Módulo de la parte servidora se utilizarán para responder a la petición del servicio Web, devolviendo los datos relativos a la consulta efectuada, y a posteriori se destruirán.
- ✓ Recuperación de roles de seguridad y definición de permisos: El servidor recupera para cada usuario validado su rol, que es el que tiene definido por su puesto en el departamento de desarrollo.
  Por otra parte, también durante el login, se suministra a cada usuario validado una lista de operaciones permitidas por role.

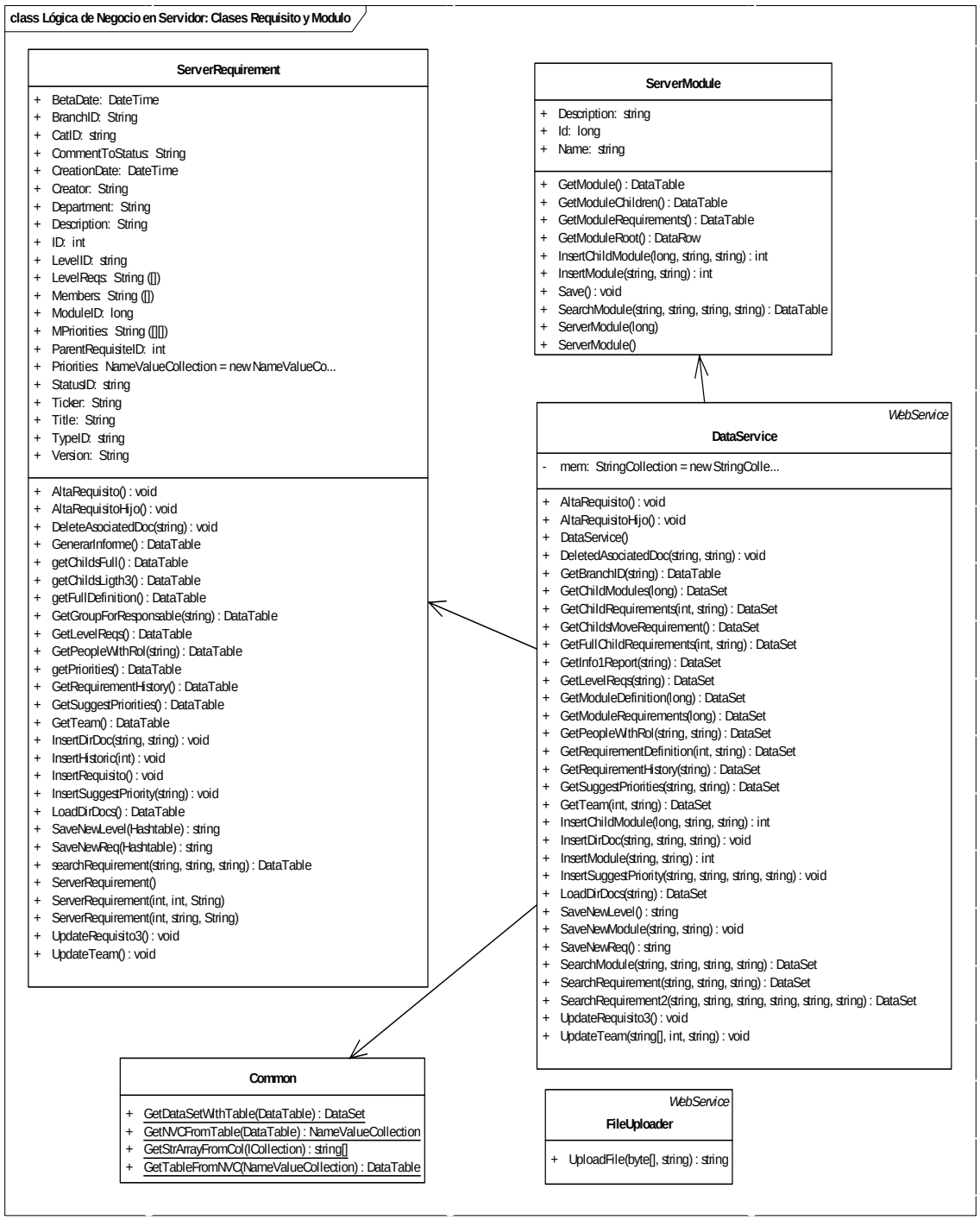
4.6.4. Modelo de clases del subsistema

Se divide el diagrama del subsistema en dos partes, las clases Requisito y Modulo de servidor junto con sus asociaciones y por otro lado la Gestión de Seguridad y Usuarios.

En la primera parte citada del subsistema se incluyen las clases réplica de la lógica de negocio en cliente, como son la clase Server Requirement y la clase Server Module. A diferencia con las clases homónimas del cliente, las clases Requirement y Module, en la parte servidora, son utilizados sus métodos, para conectar con la base de datos y llevar a cabo inserciones, consultas y modificaciones. Es decir, se accede principalmente a sus métodos y no a sus atributos. Se crea, además, una clase Common con métodos de acceso, para manejar entre otras cosas, el serializar *Datatables*.

En la segunda parte citada del subsistema, se crean las clases necesarias para la gestión de usuarios y la seguridad. La clase User permite almacenar los datos relativos a los usuarios del sistema, relacionándose con la clase User Manager, que incluye métodos para la creación de usuarios, modificación etc. La clase User se relaciona con las clases Rol y Operation para almacenar las operaciones de acceso al sistema de cada usuario.

Se muestra el Modelo de Clases del subsistema de lógica de negocio en servidor, en la página siguiente, dividido en las dos partes citadas.


class Lógica de Negocio en Servidor: Clases Requisito y Modulo
ServerRequirement
+ BetaDate: DateTime
+ BranchID: String
+ CatID: string
+ CommentToStatus: String
+ CreationDate: DateTime
+ Creator: String
+ Department: String
+ Description: String
+ ID: int
+ LevelID: string
+ LevelReqs: String ([])
+ Members: String ([])
+ ModuleID: long
+ MPriorities: String ([][])
+ ParentRequisiteID: int
+ Priorities: NameValueCollection = new NameValueCo...
+ StatusID: string
+ Ticker: String
+ Title: String
+ TypeID: string
+ Version: String
+ AltaRequisito() : void
+ AltaRequisitoHijo() : void
+ DeleteAsociatedDoc(string) : void
+ GenerarInforme() : DataTable
+ getChildsFull() : DataTable
+ getChildsLigth3() : DataTable
+ getFullDefinition() : DataTable
+ GetGroupForResponsable(string) : DataTable
+ GetLevelReqs() : DataTable
+ GetPeopleWithRol(string) : DataTable
+ getPriorities() : DataTable
+ GetRequirementHistory() : DataTable
+ GetSuggestPriorities() : DataTable
+ GetTeam() : DataTable
+ InsertDirDoc(string, string) : void
+ InsertHistoric(int) : void
+ InsertRequisito() : void
+ InsertSuggestPriority(string) : void
+ LoadDirDocs() : DataTable
+ SaveNewLevel(Hashtable) : string
+ SaveNewReq(Hashtable) : string
+ searchRequirement(string, string, string) : DataTable
+ ServerRequirement()
+ ServerRequirement(int, int, String)
+ ServerRequirement(int, string, String)
+ UpdateRequisito3() : void
+ UpdateTeam() : void
ServerModule
+ Description: string
+ Id: long
+ Name: string
+ GetModule() : DataTable
+ GetModuleChildren() : DataTable
+ GetModuleRequirements() : DataTable
+ GetModuleRoot() : DataRow
+ InsertChildModule(long, string, string) : int
+ InsertModule(string, string) : int
+ Save() : void
+ SearchModule(string, string, string, string) : DataTable
+ ServerModule(long)
+ ServerModule()
WebService
DataService
- mem: StringCollection = new StringColle...
+ AltaRequisito() : void
+ AltaRequisitoHijo() : void
+ DataService()
+ DeletedAsociatedDoc(string, string) : void
+ GetBranchID(string) : DataTable
+ GetChildModules(long) : DataSet
+ GetChildRequirements(int, string) : DataSet
+ GetChildsMoveRequirement() : DataSet
+ GetFullChildRequirements(int, string) : DataSet
+ GetInfo1Report(string) : DataSet
+ GetLevelReqs(string) : DataSet
+ GetModuleDefinition(long) : DataSet
+ GetModuleRequirements(long) : DataSet
+ GetPeopleWithRol(string, string) : DataSet
+ GetRequirementDefinition(int, string) : DataSet
+ GetRequirementHistory(string) : DataSet
+ GetSuggestPriorities(string, string) : DataSet
+ GetTeam(int, string) : DataSet
+ InsertChildModule(long, string, string) : int
+ InsertDirDoc(string, string, string) : void
+ InsertModule(string, string) : int
+ InsertSuggestPriority(string, string, string, string) : void
+ LoadDirDocs(string) : DataSet
+ SaveNewLevel() : string
+ SaveNewModule(string, string) : void
+ SaveNewReq() : string
+ SearchModule(string, string, string, string) : DataSet
+ SearchRequirement(string, string, string) : DataSet
+ SearchRequirement2(string, string, string, string, string, string) : DataSet
+ UpdateRequisito3() : void
+ UpdateTeam(string[], int, string) : void
Common
+ GetDataSetWithTable(DataTable) : DataSet
+ GetNVCFromTable(DataTable) : NameValueCollection
+ GetStrArrayFromCol(ICollection) : string[]
+ GetTableFromNVC(NameValueCollection) : DataTable
WebService
FileUploader
+ UploadFile(byte[], string) : string

4.6.5. Métodos Principales

Clase: ServerRequirement, **Método**: AltaRequisito(), **Desc**: Inserción requisito nuevo

Clase: ServerRequirement, **Método**: AltaRequisitoHijo(), **Desc**: Inserción req. hijo nuevo

Clase: ServerRequirement, **Método**: GetChildsLight3(), **Desc**: Consulta datos requisito

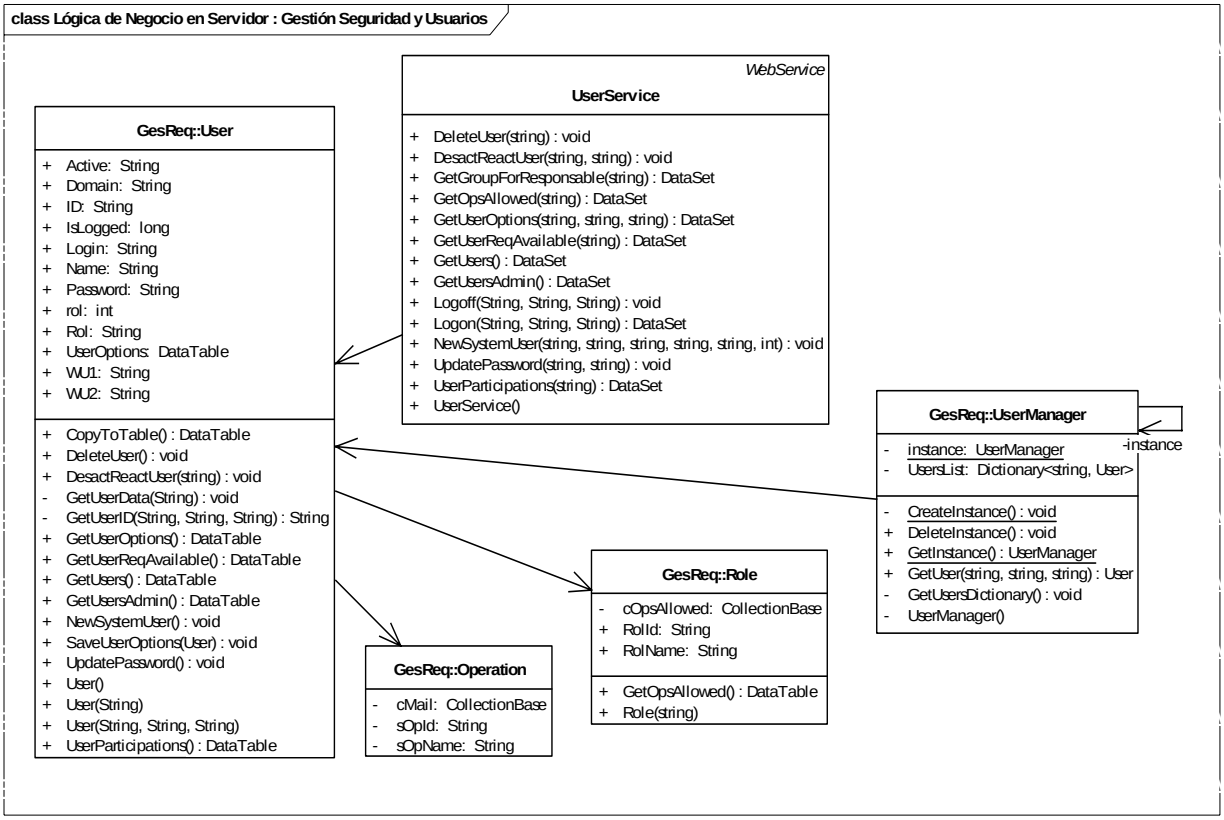
Clase: ServerRequirement, **Método**: UpdateRequisito3(), **Desc**: Modifica requisito

Clase: ServerModule, **Método**: InsertModule(), **Desc**: Inserción módulo nuevo

Clase: ServerModule, **Método**: InsertChildModule(), **Desc**: Inserción módulo hijo nuevo

Clase: ServerModule, **Método**: GetModule(), **Desc**: Consulta módulo

Clase: Common, **Método**: GetaDataSetwithTable(), **Desc**: Devolver dataset con datatable



4.6.6. Métodos Principales

Clase: User, **Método**: GetUserData(), **Desc**: Consulta datos de un usuario

Clase: User, **Método**: GetUserAdmin(), **Desc**: Consulta usuarios sistema

Clase: User, **Método**: NewSystemUser(), **Desc**: Inserción nuevo usuario sistema

Clase: UserManager, **Método**: GetUsersDictionary(), **Desc**: Consulta dicc. usuarios

Clase: Rol, **Método**: GetOpsAllowed(), **Desc**: Consulta opciones del rol

4.7. SUBSISTEMA DE PRESENTACIÓN

4.7.1. Descripción

El subsistema de presentación, como ya se adelanto en la parte de análisis, se implementará en la medida de lo posible, de acuerdo al patrón Modelo-Vista-Controlador (MVC), intentando que la interfaz sea lo menos dependiente posible de los datos.

4.7.2. Componentes básicos

- Clases controladoras:
  - Cada formulario que se implemente, llevará asociado una clase controladora, que será la clase intermediaria entre la lógica de negocio en cliente, y los propios formularios.
- Clases formulario:
  - Cada formulario, es una clase. En el caso de los formularios con elevada complejidad, se relacionarán varios controladores (ejemplo: formulario principal)

4.7.3. Detalles de implementación

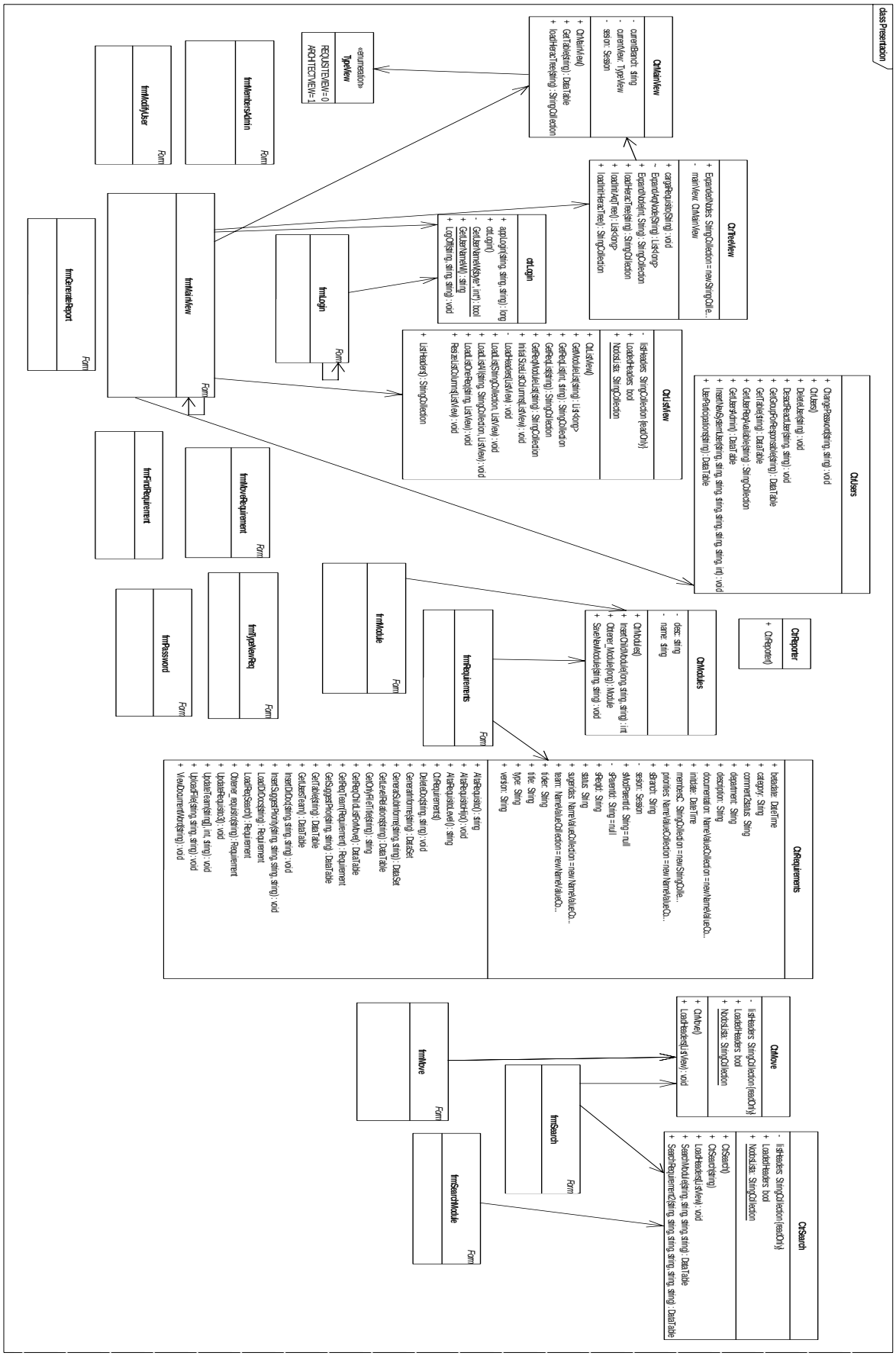
- ✓ Los detalles de implementación ya se han hecho referencia anteriormente en esta memoria. Las clases formulario, utilizan los métodos de las clases controladoras para realizar las operaciones con el resto de clases. El objetivo es hacer que la interfaz no se acople al sistema, y por tanto en caso de tener que efectuar cambios en la misma, conseguir que la realización del cambio sea lo menos costosa posible.

4.7.4. Modelo de clases del subsistema

En el Modelo de Clases del Subsistema de Presentación, se representan las clases pertenecientes, en primer lugar a la interfaz del Sistema Gestión de Requisitos, compuesta por todas las clases formulario. En segundo lugar, se representan las clases controladoras que se relacionan con los diferentes formularios creados para representar el *front end*, siendo éste un subsistema con gran importancia, dado que el conjunto de clases representadas permiten la interacción del usuario con el sistema. Por regla general, cada uno de los formularios con gran cantidad de información a introducir o a mostrar tiene una relación con una clase controladora. La clase controladora es la encargada de comunicarse con el Subsistema de Lógica de Negocio en cliente. Con esta clase conseguimos aislar la interfaz de usuario de los datos, con el consiguiente beneficio posterior, en caso de querer implementar una interfaz diferente en una posible evolución futura del sistema.

Respecto al uso del patrón MVC, comentado en la fase de análisis, para la adopción de las consideraciones de dicho patrón, se crean precisamente en relación a lo comentado anteriormente, una serie de clases controladoras. Estas clases controladoras encarnan la parte del modelo correspondiente al *CONTROLADOR*. La *VISTA* estaría simbolizada por todas las clases formulario, recordando que cada pantalla de la interfaz gráfica (descrita en el próximo punto), se implementa como una clase formulario. La utilización de las clases citadas, controladores y formularios, permite separar los datos de la presentación. Por último, el *MODELO* lo representarían las clases del negocio tanto de servidor como de cliente.

A continuación, se muestra el Modelo de Clases del Subsistema de Presentación, en la página siguiente.


class Presentation
CtlUsers
CtlReporter
CtlReporter()
CtlModules
CtlModules()
CtlRequirements
CtlMove
CtlMove()
CtlSearch
CtlSearch()
CtlMainView
CtlMainView()
CtlTreeView
CtlTreeView()
ctlLogin
ctlLogin()
CtlListView
CtlListView()
«enumeration»
TypeView
REQUISITEVIEW = 0
ARCHITECTVIEW = 1
frmMainView
frmLogin
frmModule
frmRequirements
frmMove
frmSearch
frmSearchModule
frmMoveRequirement
frmFindRequirement
frmTypeNewReq
frmPassword
frmGenerateReport
frmModifyUser
frmMembersAdmin
Form

4.7.5. Métodos Principales

Clase: ctrRequirements, **Método**: AltaRequisito(), **Desc**: Creación requisito

Clase: ctrRequirements, **Método**: AltaRequisitoHijo(), **Desc**: Creación requisito hijo

Clase: ctrRequirements, **Método**: LoadReqSearch(), **Desc**: Carga requisitos de busq.

Clase: ctrRequirements, **Método**: GetUsersTeam(), **Desc**: Consulta participantes req.

Clase: ctrSearch, **Método**: SearchModule(), **Desc**: Búsqueda de modulo

Clase: ctrSearch, **Método**: SearchRequirement(), **Desc**: Búsqueda de requisito

Clase: ctrModule, **Método**: SaveNewModule(), **Desc**: Creación módulo

Clase: ctrTreeView, **Método**: LoadInitArchTree(), **Desc**: Carga árbol arquitectura.

Clase: ctrTreeView, **Método**: LoadInitHieracTree(), **Desc**: Carga árbol jerárquico.

Clase: ctrLogin, **Método**: appLogin(), **Desc**: Login en la aplicación

Clase: ctrLogin, **Método**: LoggOff (), **Desc**: Deslogarse de la aplicación.

Clase: ctrUsers, **Método**: GetUsersAdmin(), **Desc**: Consultar usuarios sistema.

Clase: ctrListView, **Método**: LoadList(), **Desc**: Carga ListView requisitos.

4.8. LA INTERFAZ GRÁFICA

En el presente apartado, se va a realizar una aproximación a la interfaz desarrollada para el Sistema Gestión de Requisitos. La interfaz gráfica o interfaz de usuario, es el medio a través del cual el usuario puede comunicarse con la computadora, equipo o dispositivo. El fin de toda interfaz es ser lo más simple posible y amigable para el usuario. Hay diferentes tipos de interfaces de usuario, y en este caso se va a describir y mostrar la interfaz gráfica del Sistema Gestión de Requisitos.

- **Pantalla Principal**

  La pantalla principal, será la pantalla de bienvenida tras acceder al sistema haciendo *login* con un usuario y password válidos, es decir que ya hayan sido registrados previamente en el sistema.

  En la parte de la izquierda se representan los requisitos de la *versión de desarrollo* que esté seleccionada en ese momento, en el combo de la parte superior. Al acceder a la aplicación estará seleccionada la *versión de desarrollo* seleccionada en el momento de salir del sistema la última vez que el usuario accedió.

  En la parte central/derecha aparecerán, si se selecciona uno de los requisitos del árbol de la parte izquierda (si está seleccionada la vista jerárquica), el propio requisito junto con los posibles hijos de ese requisito, en caso de que fuera un requisito padre, y no un requisito hoja (realizando la equivalencia con los nodos hoja de los árboles de representación).

  En la parte inferior, se ubican las pestañas que permiten alternar entre la vista de componentes de arquitectura, y la propia vista jerárquica.

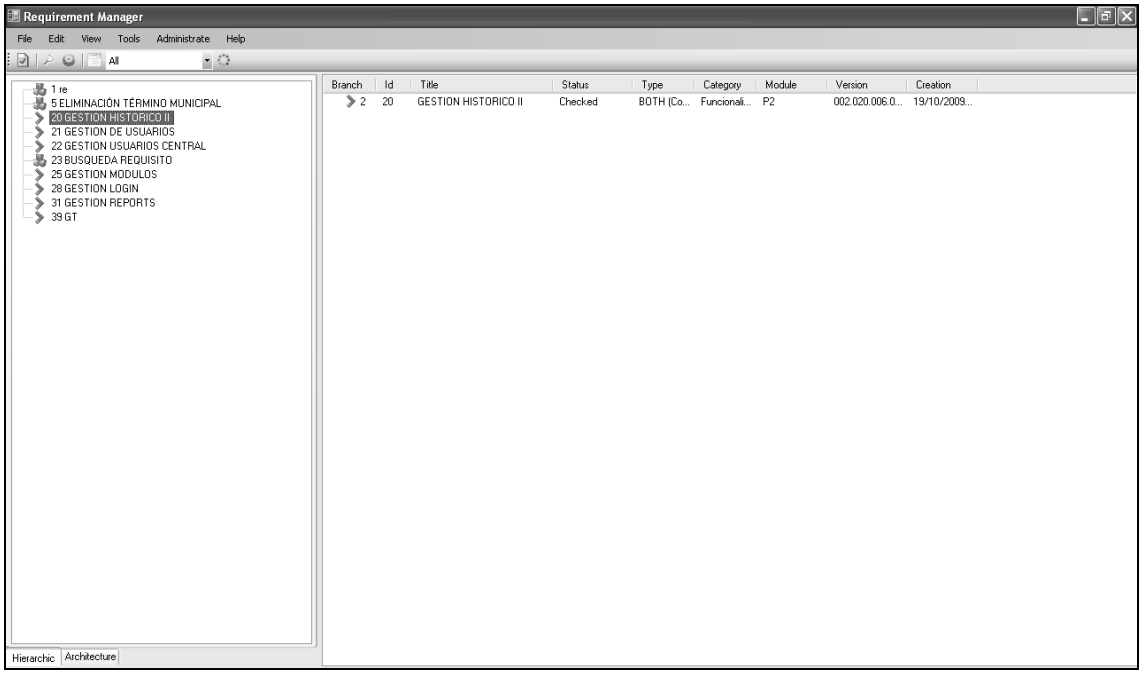
  Para acceder a la información referente a un requisito, se debe hacer clic sobre cualquiera de los requisitos que se muestren en la parte central/derecha.

- **Pantalla Requisito**

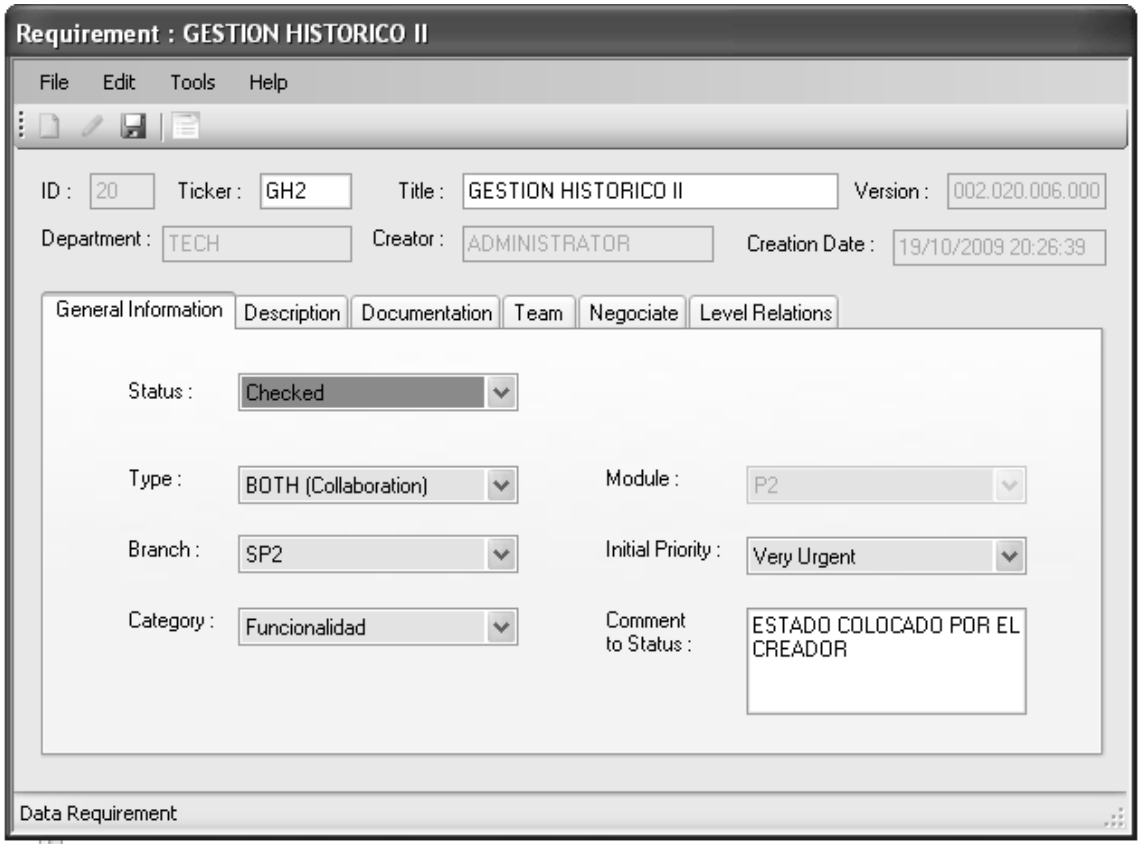
  La pantalla requisito, es otra de las pantallas esenciales del sistema. En ella se muestra toda la información referente al requisito. Se utiliza la misma pantalla, tanto para la creación de requisitos, la modificación y la consulta. En la parte superior se ubican los iconos con las

funcionalidades posibles. La información se divide en pestañas, y cada una de ellas hace referencia a una tarea dentro de los diferentes estados que puede tener un requisito, por ejemplo, la pestaña “Team” es la pestaña dónde se realizaría la asignación de participantes, justo cuando el requisito en estado *ABIERTO* pasa a estado *CHEQUEADO*.

A continuación se muestran las pantallas, implementados los formularios asociados a cada una de las descripciones anteriores:



Pantalla Principal



Pantalla Requisito

5. ORGANIZACIÓN DEL PROYECTO

5.1. CONCEPTO DE PROYECTO

El trabajo a través de proyectos, es la forma habitual de actuación en el desarrollo de software. La decisión de emprender un proyecto suele ser consecuencia de distintas circunstancias y situaciones.

Podemos definir Proyecto, como:

> PROYECTO: *Conjunto de etapas, actividades y tareas que tiene como finalidad alcanzar un objetivo, que implica un trabajo no inmediato, a un plazo relativamente largo.* BIBLIOGRAFÍA [1]

En general, un proyecto tiene las siguientes características. BIBLIOGRAFÍA [11]

- Implica un principio y un final
- Utiliza diversos recursos finitos y cuenta con un presupuesto
- Tiene actividades únicas y generalmente no repetitivas
- Tiene un objetivo
- Requiere un jefe de proyecto y personal de desarrollo cuyos roles y estructura de equipo deben definirse y desarrollarse
- Tiene que planificarse

En apartados siguientes, se definirán estas características para el proyecto sistema Gestión de Requisitos.

5.2. MODELO DEL CICLO DE VIDA DEL SOFTWARE

El estándar ISO/IEC 12207-1 entiende por Modelo del Ciclo de Vida, un marco de referencia que contiene los procesos, las tareas, actividades involucradas en el desarrollo, explotación y ampliación de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta su final de uso. El estándar IEE1074 establece que es una aproximación a la adquisición suministro, desarrollo, explotación y mantenimiento de un producto software. Ambas consideran una actividad como un conjunto de tareas, y una tarea como una acción que transforma entradas en salidas. Conviene destacar que a diferencia de los Modelos de Procesos, los Modelos del Ciclo de vida, no hablan de procesos que se deben de realizar para desarrollar el software, sino que actividades y tareas , y en qué orden deben de llevarse a cabo.

En resumen, un MODELO DEL CICLO DE VIDA DEL SOFTWARE:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

El modelo del ciclo de vida escogido, para el desarrollo del sistema, se ha elegido antes de entrar en la fase de análisis del mismo, permitiendo así tener desde el primer momento, una idea clara de cómo se iba a efectuar el desarrollo. Observando las dimensiones del alcance del proyecto, y que esas dimensiones serían amplias se decide escoger el de modelo de “Aproximación incremental”.

5.2.1. Modelo de Aproximación incremental

Los riesgos asociados con el desarrollo de sistemas largos y complejos son enormes. Una forma de reducir los riesgos es construir sólo una parte del sistema, reservando otros aspectos para niveles posteriores. El desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema. Típicamente, un documento de requerimientos es escrito al capturar todos los requerimientos para el sistema completo.

Note que el desarrollo incremental es 100% compatible con el modelo cascada. El desarrollo incremental no demanda una forma específica de observar el desarrollo de algún otro incremento. Así, el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo.

El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
- Si un error importante es realizado, sólo la última iteración necesita ser descartada.
- Reduciendo el tiempo de desarrollo de un sistema (en este caso en incremento del sistema) decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.
- Si un error importante es realizado, el incremento previo puede ser usado.
- Los errores de desarrollo realizados en un incremento, pueden ser arreglados antes del comienzo del próximo incremento.

Un ejemplo de este paradigma se tiene en el desarrollo de una aplicación sencilla, como es un editor de textos. En el primer incremento se podría desarrollar con un reducido conjunto de funciones, como las funciones básicas de gestión de archivos. En un segundo incremento, se puede incluir la gestión avanzada de textos. Y en un tercer incremento se pondría la corrección ortográfica.

5.3. PLANIFICACIÓN DEL PROYECTO

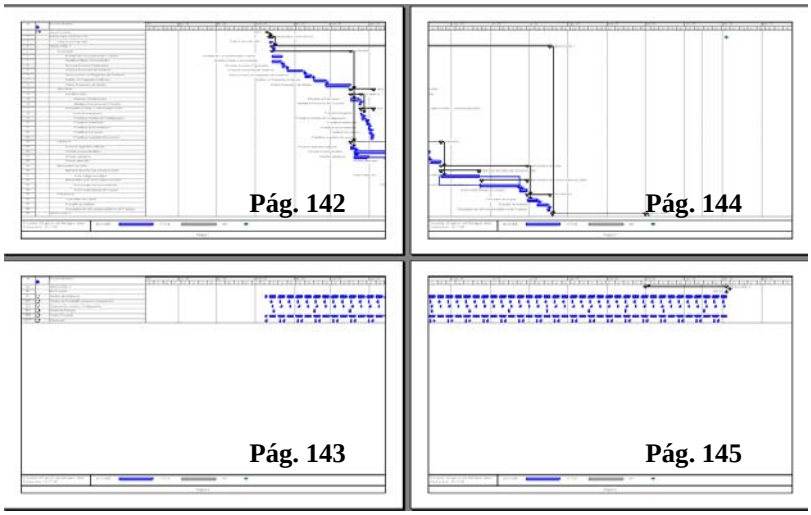
Uno de los aspectos fundamentales en los proyectos es la planificación del mismo. Se efectúa una planificación del proyecto Sistema de Gestión de Requisitos mediante la herramienta Microsoft Project.

5.3.1. Diagrama de Gantt

El diagrama de Gantt es una herramienta de planificación, generalmente utilizada en proyectos de pequeña envergadura, es decir, sin un gran número de actividades. Es un diagrama de barras, en forma de tabla, donde se hace una referencia cruzada entre las tareas (filas) y los tiempos de ejecución de las mismas (columnas). Dentro del diagrama se pueden incluir fases, que engloben dichas tareas. Además, el diagrama de Gantt permite definir el calendario, y las herramientas que posibilitan la generación de diagramas de Gantt, suelen permitir introducir la precedencia entre tareas y la asignación de recursos a las mismas.

En las páginas siguientes se representa el diagrama de Gannt del proyecto Sistema Gestión de Requisitos.

Debido al tamaño del Diagrama de Gantt, en la siguiente figura se identifica cada parte del diagrama, con la página dónde se encuentra:

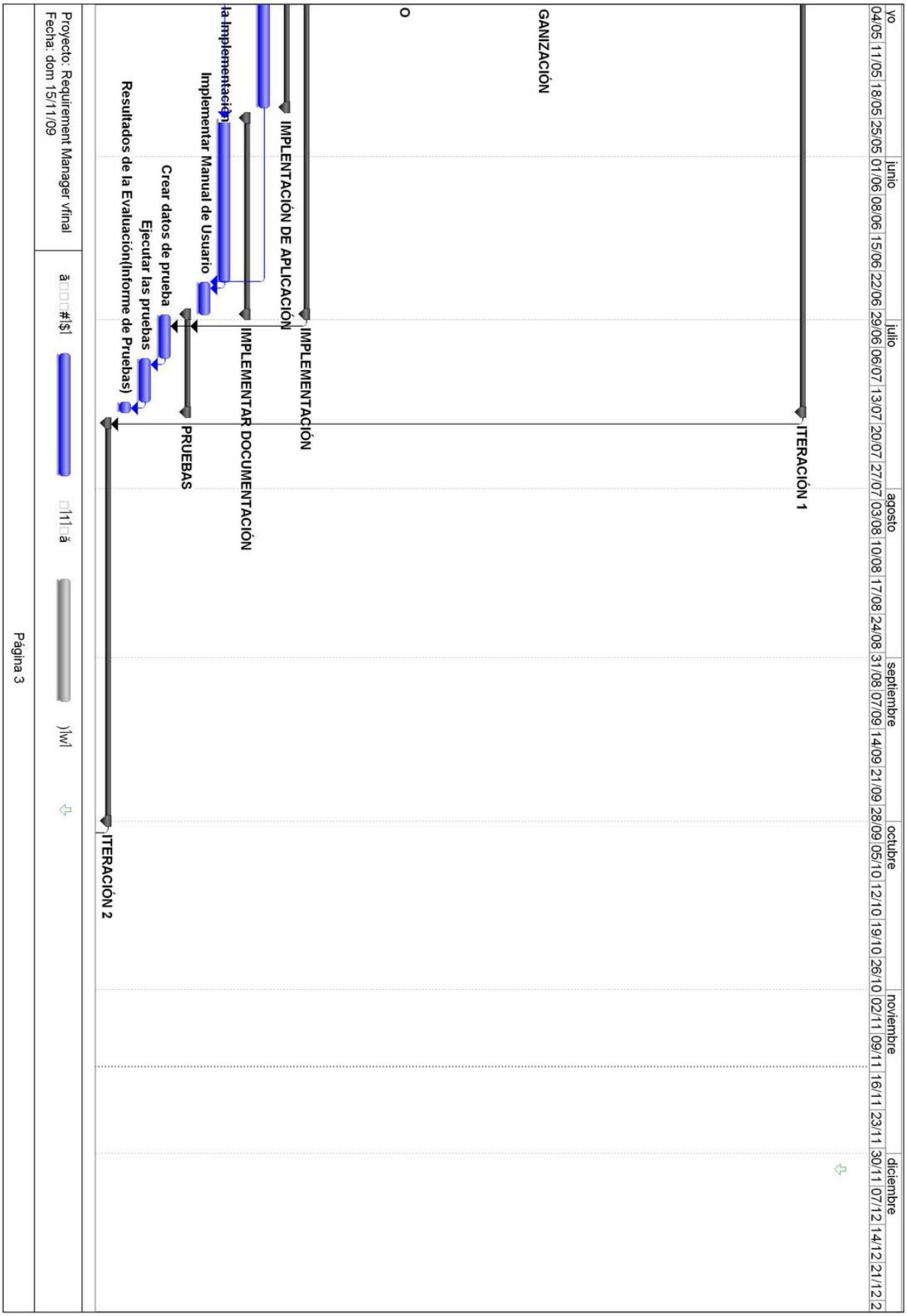


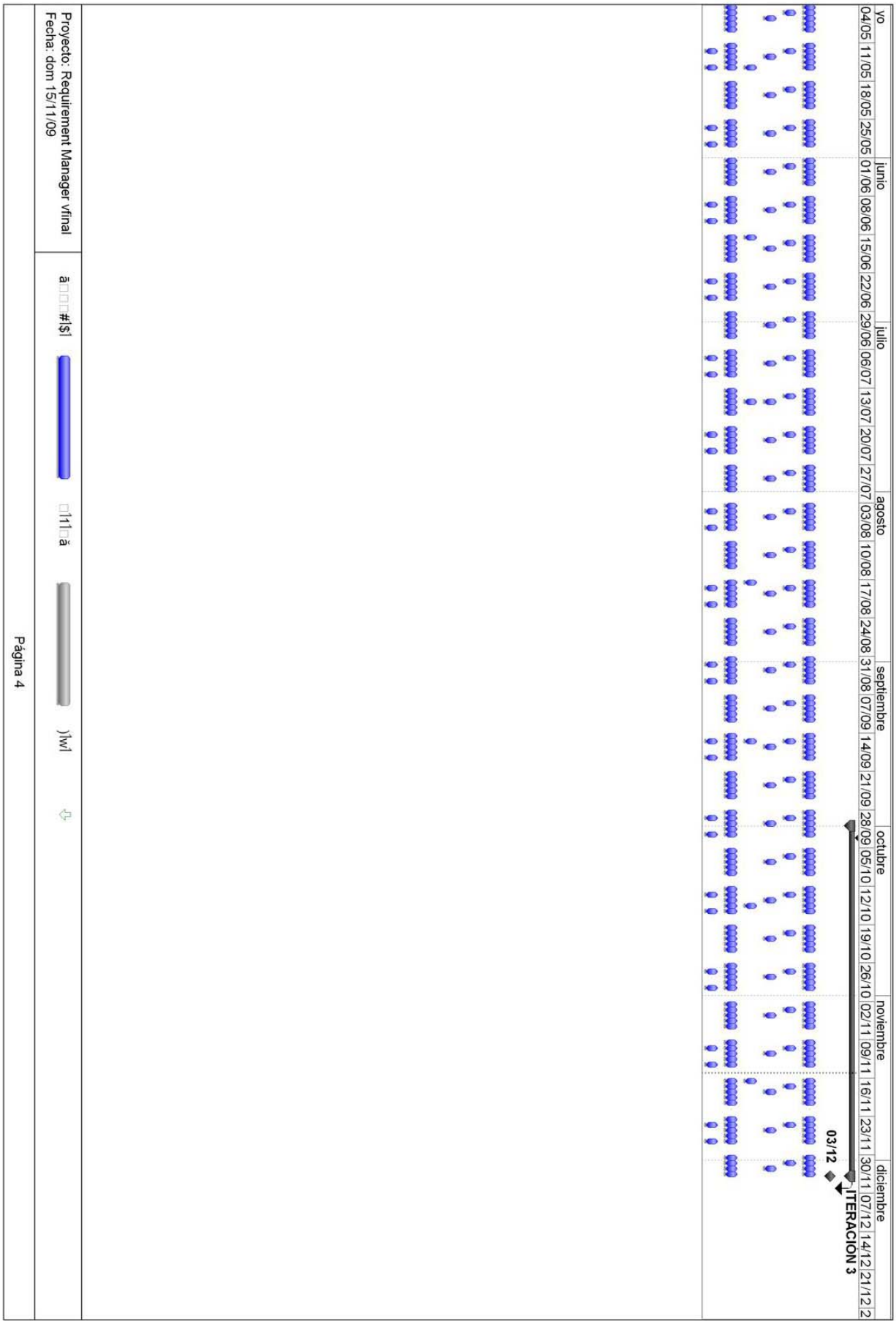
Id
Nombre de tarea
diciembre
enero
febrero
marzo
abril
mayo
01/12 08/12 15/12 22/12 29/12 05/01 12/01 19/01 26/01 02/02 09/02 16/02 23/02 02/03 09/03 16/03 23/03 30/03 06/04 13/04 20/04 27/04 04/
1 Inicio Proyecto
2 INICIO DEL PROYECTO
3 Crear el ciclo de vida
4 ITERACIÓN 1
5 ANÁLISIS
6 Estudio de Comportamientos Viables
7 Identificar Ideas o Necesidades
8 Formular Accesos Potenciales
9 Analizar Funciones del Sistema
10 Descomponer los Requisitos del Sistema
11 Definir los Requisitos Software
12 Definir Requisitos de Interfaz
13 GESTIÓN
14 ESTIMACIÓN
15 Efectuar la Estimación
16 Identificar Recursos del Proyecto
17 PLANIFICACIÓN Y ORGANIZACIÓN
18 Plan de evaluación
19 Planificar Gestión de Configuración
20 Planificar Instalación
21 Planificar documentación
22 Planificar Formación
23 Planificar la gestión de proyecto
24 DISEÑO
25 Priorizar requisitos software
26 Diseñar la base de datos
27 Diseñar interfaces
28 Diseño detallado
29 IMPLEMENTACIÓN
30 IMPLENTACIÓN DE APLICACIÓN
31 Crear código ejecutable
32 IMPLEMENTAR DOCUMENTACIÓN
33 Documentar la Implementación
34 Implementar Manual de Usuario
35 PRUEBAS
36 Crear datos de prueba
37 Ejecutar las pruebas
38 Resultados de la Evaluación(Informe de Pruebas
39 ITERACIÓN 2
09/01
Proyecto: Requirement Manager vfinal
Fecha: dom 15/11/09
Página 1

Id	Nombre de tarea
64	**ITERACIÓN 3**
90	Fin Proyecto
91	**Gestión de Históricos**
327	**Gestión de Calidad(Evaluación e Instalación)**
375	**Gestionar Formación y Configuración**
423	**Gestionar Riesgos**
435	**Control Proyecto**
671	**Reuniones**

diciembre | enero | febrero | marzo | abril | mayo
01/12 | 08/12 | 15/12 | 22/12 | 29/12 | 05/01 | 12/01 | 19/01 | 26/01 | 02/02 | 09/02 | 16/02 | 23/02 | 02/03 | 09/03 | 16/03 | 23/03 | 30/03 | 06/04 | 13/04 | 20/04 | 27/04 | 04/

Proyecto: Requirement Manager vfinal
Fecha: dom 15/11/09


yo
junio
julio
agosto
septiembre
octubre
noviembre
diciembre
04/05 11/05 18/05 25/05 01/06 08/06 15/06 22/06 29/06 06/07 13/07 20/07 27/07 03/08 10/08 17/08 24/08 31/08 07/09 14/09 21/09 28/09 05/10 12/10 19/10 26/10 02/11 09/11 16/11 23/11 30/11 07/12 14/12 21/12 2
GANIZACIÓN
O
ITERACIÓN 1
IMPLENTACIÓN DE APLICACIÓN
IMPLEMENTACIÓN
la Implementación
Implementar Manual de Usuario
IMPLEMENTAR DOCUMENTACIÓN
Crear datos de prueba
Ejecutar las pruebas
Resultados de la Evaluación(Informe de Pruebas)
PRUEBAS
ITERACIÓN 2
Proyecto: Requirement Manager vfinal
Fecha: dom 15/11/09
Página 3


Proyecto: Requirement Manager vfinal
Fecha: dom 15/11/09
mayo
04/05 11/05 18/05 25/05
junio
01/06 08/06 15/06 22/06 29/06
julio
06/07 13/07 20/07 27/07
agosto
03/08 10/08 17/08 24/08 31/08
septiembre
07/09 14/09 21/09 28/09
octubre
05/10 12/10 19/10 26/10
noviembre
02/11 09/11 16/11 23/11 30/11
diciembre
07/12 14/12 21/12
03/12
ITERACIÓN 3
Página 4

5.3.2. Recursos

Todo proyecto, para su desarrollo, necesita recursos, y esos recursos son obviamente recursos finitos, deben de identificarse y estar acordes a un presupuesto.

Se van a distinguir dos tipos de recursos necesarios para la ejecución del proyecto Sistema Gestión de Requisitos:

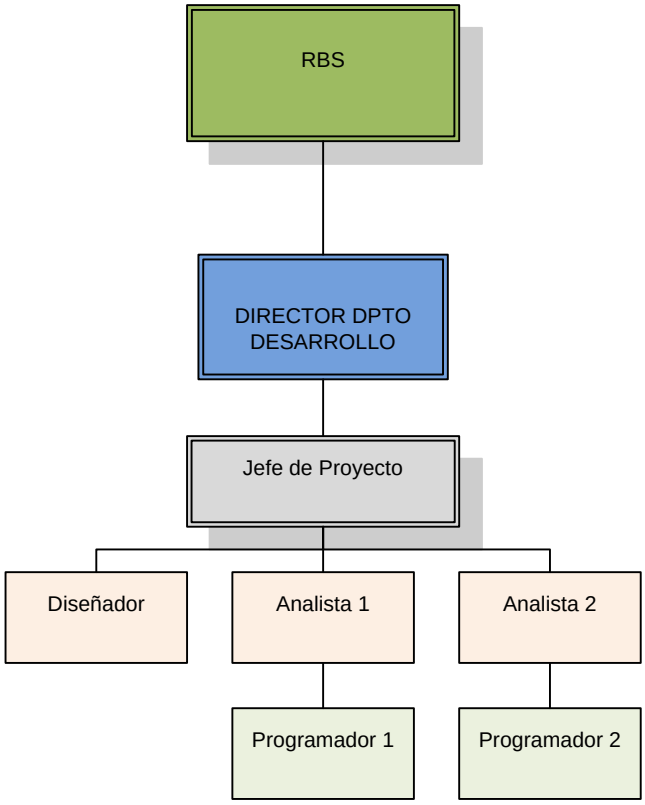
- Recursos humanos: Personas que participan en el proyecto.
- Recursos materiales: Recursos tangibles necesarios.

Para representar los recursos, se utiliza el diagrama RBS (Resources Breakdown Structure). Se representan en un diagrama, los recursos humanos involucrados, y en otro diagrama, los recursos materiales necesarios.

Objetivo del RBS de RRHH:

Mostrar gráficamente la organización humana del proyecto.

- RBS (Recursos humanos)



RBS: Recursos humanos

Respecto al diagrama anterior, se comentan los diferentes perfiles de los recursos humanos del proyecto:

- Director del Dpto. de Desarrollo:

  Perteneciente al nivel ejecutivo. Debe de conocer la existencia del proyecto, y efectuar seguimientos sobre el mismo. Responsable global de los proyectos que se realicen en su área.

- Jefe de Proyecto

  Responsable de controlar, planificar y dirigir las actividades del proyecto. Responsable del mismo frente al Director del Dpto.

  Coste (euros/hora): 40 €/h

- Analista

  Responsable de la parte de análisis del proyecto. Debe identificar funcionalidades y requisitos del sistema, y coordinar la implementación del mismo.

  Coste (euros/hora): 30 €/h

- Diseñador

  Responsable de la interfaz del sistema. Desarrolla su trabajo en paralelo a los analistas.

  Coste (euros/hora): 25 €/h

- Programador

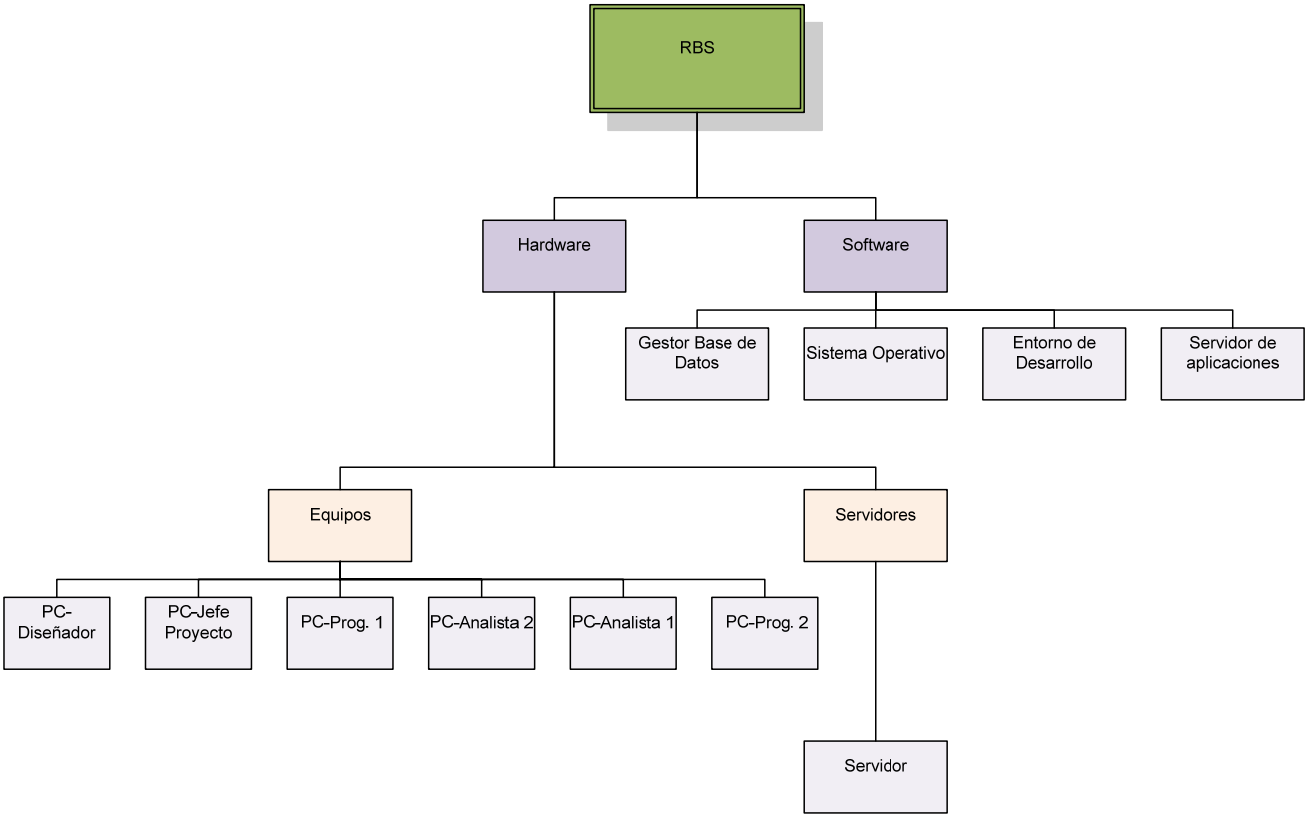
  Responsable de la implementación del sistema, de acuerdo al análisis efectuado por los analistas.

  Coste (euros/hora): 20 €/h

Objetivo del RBS de Materiales:

Reflejar la estructura de recursos materiales necesarios para la realización del proyecto.

- RBS (recursos materiales)



RBS : Recursos materiales

Respecto al diagrama anterior, se comenta la necesidad de proveer a cada integrante del equipo del proyecto de un equipo informático para realizar su trabajo. Para la puesta en producción del sistema serán necesarios dos servidores, uno para la parte servidora de la aplicación y opcionalmente uno para la base de datos. Respecto al software, será necesario disponer de las licencias del sistema operativo (único en todos los equipos), del gestor de base de datos y del entorno de desarrollo.

RECURSO	UNIDADES	PRECIO UNIDAD	PRECIO TOTAL
HARDWARE			
PORTÁTIL INTEL CORE 2 DUO	6	600	3600
SERVIDOR IBM	1	1200	1200
SOFTWARE			
LICENCIA WINDOWS XP	6	200	1200
LICENCIA SGBD SQL SERVER 2005	5	150	900
LICENCIA VISUAL STUDIO 2008	4	200	800
Internet Information Server (IIS)	1	0	0
		TOTAL	7700

Importe expresados en euros (€)

5.3.3. Costes

En este apartado, se realiza un cálculo de los costes del proyecto. El cálculo se basa en la asignación de los recursos humanos a las diferentes tareas del proyecto:

Recursos	**Tipo**	**Datos**	**Total**
Analista1	Trabajo	Costo	20760
		Trabajo	692
Costo Analista1			20760
Trabajo Analista1			692
Analista2	Trabajo	Costo	20760
		Trabajo	692
Costo Analista2			20760
Trabajo Analista2			692
Diseñador	Trabajo	Costo	11600
		Trabajo	464
Costo Diseñador			11600
Trabajo Diseñador			464
Jefe de Proyecto	Trabajo	Costo	7200
		Trabajo	240
Costo Jefe de Proyecto			9600
Trabajo Jefe de Proyecto			240
Programador1	Trabajo	Costo	13200
		Trabajo	660
Costo Programador1			13200
Trabajo Programador1			660
Programador2	Trabajo	Costo	13200
		Trabajo	660
Costo Programador2			13200
Trabajo Programador2			660
Sin asignar	Trabajo	Costo	0
		Trabajo	0
Total Costo			89120
Total Trabajo			3408

Costo = Euros;
Trabajo = Horas;

Proyecto SGR	Iteración	Tareas	Actividades	Subactividades	Costo
	Inicio Proyecto				0
	INICIO DEL PROYECTO	Crear el ciclo de vida			640
	Total INICIO DEL PROYECTO				640
	ITERACIÓN 1	ANÁLISIS	Estudio de Comportamientos Viables		1440
			Identificar Ideas o Necesidades		1200
			Formular Accesos Potenciales		960
			Analizar Funciones del Sistema		4800
			Descomponer los Requisitos del Sistema		2400
			Definir los Requisitos Software		3360
			Definir Requisitos de Interfaz		4760
		Total ANÁLISIS			18920
		GESTIÓN	ESTIMACIÓN	Efectuar la Estimación	1600
				Identificar Recursos del Proyecto	320
			Total ESTIMACIÓN		1920
			PLANIFICACIÓN Y ORGANIZACIÓN	Plan de evaluación	320
				Planificar Gestión de Configuración	640
				Planificar Instalación	320
				Planificar documentación	320
				Planificar Formación	320
				Planificar la gestión de proyecto	640
			Total PLANIFICACIÓN Y ORGANIZACIÓN		2560
		Total GESTIÓN			4480
		DISEÑO	Priorizar requisitos software		1320
			Diseñar la base de datos		3400
			Diseñar interfaces		1400
			Diseño detallado		1400
		Total DISEÑO			7520
		IMPLEMENTACIÓN	IMPLENTACIÓN DE APLICACIÓN	Crear código ejecutable	7040
			Total IMPLENTACIÓN DE APLICACIÓN		7040

			IMPLEMENTAR DOCUMENTACIÓN	Documentar la Implementación	3520
				Implementar Manual de Usuario	1280
			Total IMPLEMENTAR DOCUMENTACIÓN		4800
		Total IMPLEMENTACIÓN			11840
		PRUEBAS	Crear datos de prueba		1920
			Ejecutar las pruebas		2040
			Resultados de la Evaluación(Informe de Pruebas)		640
		Total PRUEBAS			4600
	Total ITERACIÓN 1				47360
	ITERACIÓN 2	ANÁLISIS II	Analizar Funciones del Sistema II		2400
			Descomponer los Requisitos del Sistema II		960
			Definir los Requisitos Software II		1920
			Definir Requisitos de Interfaz II		2720
		Total ANÁLISIS II			8000
		GESTIÓN II	ESTIMACIÓN	Efectuar la Estimación II	960
				Identificar Recursos del Proyecto II	640
			Total ESTIMACIÓN		1600
		Total GESTIÓN II			1600
		DISEÑO II	Priorizar requisitos software II		1320
			Diseñar la base de datos II		1360
			Diseñar interfaces II		800
			Diseño detallado II		600
		Total DISEÑO II			4080
		IMPLEMENTACIÓN II	IMPLENTACIÓN DE APLICACIÓN	Crear código ejecutable	1920
			Total IMPLENTACIÓN DE APLICACIÓN		1920
			IMPLEMENTAR DOCUMENTACIÓN	Documentar la Implementación	1920
				Implementar Manual de Usuario	640
			Total IMPLEMENTAR DOCUMENTACIÓN		2560
		Total IMPLEMENTACIÓN II			4480
		PRUEBAS II	Crear datos de prueba II		1600

		Ejecutar las pruebas II		2400
		Resultados de la Evaluación(Informe de Pruebas) II		320
	Total PRUEBAS II			4320
Total ITERACIÓN 2				22480
ITERACIÓN 3	ANÁLISIS III	Analizar Funciones del Sistema III		1440
		Descomponer los Requisitos del Sistema III		960
		Definir los Requisitos Software III		1440
		Definir Requisitos de Interfaz III		960
	Total ANÁLISIS III			4800
	GESTIÓN III	ESTIMACIÓN	Efectuar la Estimación III	960
			Identificar Recursos del Proyecto III	640
		Total ESTIMACIÓN		1600
	Total GESTIÓN III			1600
	DISEÑO III	Priorizar requisitos software III		1360
		Diseñar la base de datos III		880
		Diseñar interfaces III		600
		Diseño detallado III		600
	Total DISEÑO III			3440
	IMPLEMENTACIÓN III	IMPLENTACIÓN DE APLICACIÓN	Crear código ejecutable III	1920
		Total IMPLENTACIÓN DE APLICACIÓN		1920
		IMPLEMENTAR DOCUMENTACIÓN	Documentar la Implementación III	1920
			Implementar Manual de Usuario III	960
			Producir y Distribuir Manual de Usuario	160
		Total IMPLEMENTAR DOCUMENTACIÓN		3040
	Total IMPLEMENTACIÓN III			4960
	PRUEBAS III	Crear datos de prueba III		1280
		Ejecutar las pruebas III		1920
		Resultados de la Evaluación(Informe de Pruebas) III		640
	Total PRUEBAS III			3840
Total ITERACIÓN 3				18640

	Fin Proyecto	0
	Gestión de Históricos	0
	Gestión de Calidad(Evaluación e Instalación)	0
	Gestionar Formación y Configuración	0
	Gestionar Riesgos	0
	Control Proyecto	0
	Reuniones	0
Total general		89120

El coste total del proyecto, es la suma del coste de los recursos humanos, asignados a las tareas del proyecto, más la suma del coste de los recursos materiales, cálculo efectuado anteriormente junto con el RBS de recursos materiales.

Costes recursos materiales: Costes HW + Costes SW = 7700 €
Costes recursos humanos: 89.120 €

COSTES = 89.120 € + 7.700 € = 96.820 €
MARGEN DE BENEFICIO (12%) = 11.618’40 €
PRECIO BRUTO = 108.438’40
IVA (16%) = 17.350’14 €
COSTE TOTAL DEL PROYECTO = 125.788’54 €

6. PRUEBAS

6.1. CONCEPTOS Y DEFINICIONES

Una de las características típicas del desarrollo del software, es la realización en la gran mayoría de los desarrollos, de controles periódicos, normalmente coincidiendo con los diferentes hitos del proyecto. Los controles pretenden evaluar la calidad de los productos generados. A continuación se hace una aproximación al concepto de Calidad del Software:

> CALIDAD DEL SOFTWARE: *Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente* BIBLIOGRAFÍA [7]

En relación a esta definición se puede extraer, que los requisitos del software, son la base de las medidas de la calidad, es decir, si un producto desarrollado cumple con los requisitos previamente encontrados, se acercará más al concepto de calidad. De ahí se reseña, por tanto, que el Sistema de Gestión de Requisitos, favorece el desarrollo de software de calidad. Todo sistema desarrollado para comprobar que se aproxima a la calidad, no obstante, debe de ser probado mediante su ejecución controlada antes de ser entregado al cliente. Las ejecuciones, o ensayos de funcionamiento, posteriores a la fase de implementación del código, se denominan Pruebas.

Se define Prueba de Software, como:

> PRUEBA DE SOFTWARE (test): *Actividad en la cual, un sistema o uno de sus componentes se ejecuta, en circunstancias previamente especificadas, los resultados se observan y registran y se realiza una evaluación de algún aspecto*. BIBLIOGRAFÍA [1]

Las pruebas constituyen un método más para poder verificar y validar el software. Verificar y validar son dos conceptos diferentes:

Según la explicación informal [BIBLIOGRAFÍA 2]:

Verificación: ¿Estamos construyendo correctamente el producto?

Validación: ¿Estamos construyendo el producto correcto?

Es necesario para comprender el proceso de pruebas, aproximarse al concepto de otro concepto fundamental en el proceso de prueba, el Caso de Prueba:

> CASO DE PRUEBA (test case): *Conjunto de entradas, condiciones de ejecución, y resultados esperados desarrollados para un objetivo particular, como por ejemplo ejercitar un camino concreto de un programa.* BIBLIOGRAFÍA [1]

6.2. DEFINICIÓN CASOS DE PRUEBA

En el Sistema de Gestión de Requisitos, como se ha citado en puntos anteriores el Modelo de Ciclo de Vida escogido ha sido Aproximación Incremental. Al haber elegido este modelo, se han ido realizando pruebas según se abordaba cada fase del desarrollo.

A continuación se van a mostrar las pruebas realizadas, por medio de los diferentes Casos de Prueba:

1.1Login en la aplicación	
Validación de requisito	
RUTA DE PRUEBA	Entrada a la aplicación
OBJETIVO	Comprobación de login
PROCESO	1.- Entrar en la aplicación con un usuario y password correctos y comprobar que entramos correctamente.
RESULTADO ESPERADO	Pantalla principal del sistema

1.2 Login no permite acceso. Usuario no existe.	
Caso de error	
RUTA DE PRUEBA	Entrada a la aplicación
OBJETIVO	Comprobar que no es posible entrar a la aplicación si no usuario correcto.
PROCESO	1.- Entrar en la aplicación con un usuario que no existe. 2.- Comprobar que nos aparece un mensaje advirtiéndonos que no es posible acceder.
RESULTADO ESPERADO	Mensaje de error de usuario ó password incorrectos y usuario no accede.

1.3 Login no permite acceso. Password incorrecto.	
Caso de error	
RUTA DE PRUEBA	Entrada a la aplicación
OBJETIVO	Comprobar que no es posible entrar a la aplicación si la password introducida es incorrecta.
PROCESO	1.- Entrar en la aplicación con un usuario que no existe. 2.- Comprobar que nos aparece un mensaje advirtiéndonos que no es posible acceder.
RESULTADO ESPERADO	Pantalla principal del sistema

1.4 Login no permite acceso. Sistema no permite conexión	
Caso de error	
RUTA DE PRUEBA	Entrada a la aplicación
OBJETIVO	Comprobar que no es posible entrar a la aplicación si el sistema deniega el acceso por usuario sin permisos para entrar o bien por no tener conectividad en ese momento.
PROCESO	1.- Entrar en la aplicación con un usuario que tuviese permisos para acceder a la aplicación y actualmente no los tiene y comprobar que no se puede acceder a la misma. 2.- Comprobar que si no se puede establecer en ese momento la conexión a la herramienta tampoco se permite el acceso a la misma.
RESULTADO ESPERADO	Mensaje de error de acceso denegado y usuario no accede.

2.1Crear requisito con estado a ABIERTO	
Validación de requisito	
RUTA DE PRUEBA	Nuevo requisito desde herramienta
OBJETIVO	Comprobar que es posible crear un nuevo requisito en la aplicación.
PROCESO	1.- Añadir un nuevo requisito introduciendo los datos obligatorios en la aplicación: título, tipo, descripción, estado (que será ABIERTO), prioridad inicial, versión de desarrollo, categoría. 2.- Comprobar que se graba el nuevo requisito y que es posible recuperarlo posteriormente.
RESULTADO ESPERADO	Requisito creado

2.2 Crear requisito con estado CHEQUEADO sin team.	
Caso de error	
RUTA DE PRUEBA	Nuevo requisito desde pantalla requisito
OBJETIVO	Comprobar que no es posible crear un nuevo requisito a CHEQUEADO en la aplicación hasta que no introducimos los participantes.
PROCESO	1.- Añadir un nuevo requisito introduciendo los datos obligatorios en la aplicación: título, tipo, descripción, estado (que será CHEQUEADO), prioridad inicial, versión de desarrollo, categoría. 2.- Comprobar que al grabar nos pide el team y no graba el requisito hasta que no demos estos datos.
RESULTADO ESPERADO	Pantalla con mensaje de aviso de que falta el team.

2.3 Crear requisito con estado CHEQUEADO y sin permisos para ello	
Caso de error	
RUTA DE PRUEBA	Nuevo requisito desde pantalla requisito
OBJETIVO	Comprobar que no es posible crear un nuevo requisito a CHEQUEADO si no se tiene permisos para ello
PROCESO	1.- Añadir un nuevo requisito introduciendo los datos obligatorios en la aplicación: título, tipo, descripción, estado (que será CHEQUEADO), prioridad inicial, versión de desarrollo, categoría, y team, siendo un usuario sin permiso (si no es Product Manager o Project Manager) 2.- Comprobar que al grabar el requisito aparece un mensaje de error ya que el usuario no tiene permisos para añadir requisitos a CHEQUEADO, y no se nos deja grabar el requisito hasta que no cambiemos estado a ABIERTO.
RESULTADO ESPERADO	Mensaje de error en pantalla porque no tenemos permisos para crear requisito con ese estado.

3.1 Editar requisito y no realizar cambios	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible editar un requisito y no realizarle ningún cambio.
PROCESO	1.- Abrir un requisito cualquiera y comprobar que lo podemos volver a cerrar y la aplicación nos permite realizar la consulta de los datos del mismo sin tener que modificar ninguno de sus datos.
RESULTADO ESPERADO	Requisito abierto.

4.1 Modificar requisito	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible editar un requisito.
PROCESO	1.- Entrar a la aplicación con un usuario que tenga uno de los siguientes roles con respecto al requisito: Creator, Product, Project 2.- Abrir un requisito cualquiera y modificar uno de los siguientes campos o todos: título, tipo, descripción, 3.- Comprobar que se puede grabar el cambio. 4.- Comprobar también que se cambia la versión del requisito, y se graba una nueva versión en el histórico
RESULTADO ESPERADO	Requisito modificado.

4.2 Modificar requisito y no tener permisos	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que no es posible modificar atributos del requisito.
PROCESO	1.- Entrar a la aplicación con un usuario que tenga uno de los siguientes roles con respecto al requisito: Creator, Product, Project 2.- Abrir un requisito cualquiera y modificar uno de los siguientes campos o todos: título, tipo, descripción, 3.- Comprobar que no se puede grabar el requisito ya que la aplicación nos informa de que no tenemos permisos para realizar cualquier cambio al requisito.
RESULTADO ESPERADO	Requisito no modificado

5.1 Asociar directorios	
Validación de requisito	
RUTA DE PRUEBA	Nuevo requisito o modificar requisito
OBJETIVO	Comprobar que es posible añadir directorios de documentación al requisito.
PROCESO	1.- Entrar a la aplicación con un usuario que tenga uno de los siguientes roles con respecto al requisito: Creator, Documentator, Project, Product 2.- Abrir un requisito cualquiera ó al crear un requisito nuevo e introducir los directorios de documentación. 3.- Comprobar que podemos grabar el cambio correctamente.
RESULTADO ESPERADO	Requisito con directorios de documentación, uno o varios.

5.2 Asociar directorios y no tener permisos	
Caso de error	
RUTA DE PRUEBA	Nuevo requisito o modificar requisito
OBJETIVO	Comprobar que no es posible añadir directorios de documentación al requisito.
PROCESO	1.- Entrar a la aplicación con un usuario que tenga uno de los siguientes roles con respecto al requisito: Creator, Documentator, Project, Product 2.- Abrir un requisito cualquiera ó al crear un requisito nuevo e introducir los directorios de documentación. 3.- Comprobar que al grabar el requisito nos aparece un mensaje de error que nos advierte de que no podemos modificar el mismo porque no tenemos permisos.
RESULTADO ESPERADO	Requisito no modificado

6.1 Cambiar estado a CHEQUEADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CHEQUEADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito cualquiera y modificar su estado a CHEQUEADO. 3.- Comprobar que al grabar se nos piden los participantes. Añadir los participantes. 4.- Comprobar que se graba el requisito con el nuevo estado. 5.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado CHEQUEADO.

6.2 Cambiar estado a CHEQUEADO y no tener permiso	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CHEQUEADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito cualquiera y modificar su estado a CHEQUEADO. 3.- Comprobar que al grabar nos aparece un error ya que no es posible modificar el estado a CHEQUEADO de un requisito si no se tienen permisos.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.3 Cambiar estado a DESECHADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a DESECHADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito cualquiera y modificar su estado a CHEQUEADO. 3.- Comprobar que se graba el requisito con el nuevo estado. 4.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado DESECHADO.

6.4 Cambiar estado a DESECHADO y no tener permiso	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a DESECHADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté a uno de los siguiente estados: ABIERTO, CHEQUEADO, CODIFICADO, TESTEADO ó ANALIZADO 3.- Intentar grabar y comprobar que nos aparece un error ya que no se nos permite modificar a estado DESECHADO si el requisito no estaba en uno de los estados anteriores sino en otro.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.5 Cambiar estado a ANALIZADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ANALIZADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito cualquiera y modificar su estado a ANALIZADO. 3.- Comprobar que el sistema nos solicita los directorios de análisis y diseño, tipo de requisito y fecha de beta. 4.- Comprobar que se graba el requisito con el nuevo estado. 5.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado ANALIZADO.

6.6 Cambiar estado a ANALIZADO y el anterior no era CHEQUEADO	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ANALIZADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté CHEQUEADO y modificar su estado a ANALIZADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje de error ya que es necesario que su estado anterior fuese CHEQUEADO .
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.7 Cambiar a estado ANALIZADO y no dar directorios	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ANALIZADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté CHEQUEADO y modificar su estado a ANALIZADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje solicitándonos los directorios de documentos. 4.- Comprobar que si no se introducen directorios el requisito no se puede grabar.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.8 Cambiar estado a CODIFICADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CODIFICADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito cualquiera y modificar su estado a CODIFICADO. 3.- Comprobar que el sistema nos solicita los directorios de CODIFICADO. 4.- Comprobar que se graba el requisito con el nuevo estado. 5.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado CODIFICADO.

6.9 Cambiar estado a CODIFICADO y el anterior no era ANALIZADO	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CODIFICADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté ANALIZADO y modificar su estado a CODIFICADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje de error ya que es necesario que su estado anterior fuese ANALIZADO.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.10 Cambiar a estado CODIFICADO y no dar directorios	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CODIFICADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que esté a ANALIZADO y modificar su estado a CODIFICADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje solicitándonos los directorios de documentos. 4.- Comprobar que si no se introducen directorios el requisito no se puede grabar.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.11 Cambiar estado a TESTEADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a TESTEADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito con estada a CODIFICADO y modificar su estado a TESTEADO. 3.- Comprobar que el sistema nos solicita los directorios de TESTEADO. 4.- Comprobar que se graba el requisito con el nuevo estado. 5.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado TESTEADO.

6.12 Cambiar estado a TESTEADO y el anterior no era CODIFICADO	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a TESTEADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté CODIFICADO y modificar su estado a TESTEADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje de error ya que es necesario que su estado anterior fuese CODIFICADO.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.13 Cambiar a estado TESTEADO y no dar directorios	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a TESTEADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que esté a CODIFICADO y modificar su estado a TESTEADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje solicitándonos los directorios de documentos. 4.- Comprobar que si no se introducen directorios el requisito no se puede grabar.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.14 Cambiar estado a CERRADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CERRADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito con estada a TESTEADO y modificar su estado a CERRADO. 3.- Comprobar que se graba el requisito con el nuevo estado. 4.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado CERRADO.

6.12 Cambiar estado a CERRADO y el anterior no era TESTEADO	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CERRADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté TESTEADO y modificar su estado a CERRADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje de error ya que es necesario que su estado anterior fuese TESTEADO.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.14 Cambiar estado a APARTADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a CERRADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que esté a uno de los siguientes estados: CHEQUEADO, ANALIZADO ó CODIFICADO y modificar su estado a APARTADO. Intentar grabar. 3.- Comprobar que se graba el requisito con el nuevo estado. 4.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado APARTADO.

6.15 Cambiar a estado APARTADO y el anterior no era CHEQUEADO, ANALIZADO ó CODIFICADO	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a APARTADO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que esté a uno de los siguientes estados: CHEQUEADO, ANALIZADO ó CODIFICADO y modificar su estado a APARTADO. Intentar grabar. 3.- Comprobar que nos aparece un mensaje de error ya que es necesario que su estado anterior fuese uno de los descritos.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.14 Cambiar estado a ABIERTO	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ABIERTO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que esté a uno de los siguientes estados: CHEQUEADO, ANALIZADO ó CODIFICADO y modificar su estado a ABIERTO. Intentar grabar. 3.- Comprobar que se graba el requisito con el nuevo estado. 4.- Comprobar también que se graba el cambio en el histórico del requisito
RESULTADO ESPERADO	Requisito a estado APARTADO.

6.15 Cambiar estado a ABIERTO y el anterior no era checking	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ABIERTO.
PROCESO	1.- Entrar a la aplicación 2.- Abrir un requisito que no esté CHEQUEADO y modificar su estado a ABIERTO. Intentar grabar. 3.- Comprobar que aparece un error ya que el requisito no estaba a CHEQUEADO con lo cual no se puede modificar su estado.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

6.15 Cambiar estado a ABIERTO y no tener permisos	
Caso de error	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que es posible cambiar el estado de un requisito a ABIERTO.
PROCESO	1.- Entrar a la aplicación con un usuario cuyo role sea distinto a product o product manager. 2.- Abrir un requisito que esté CHEQUEADO y modificar su estado a ABIERTO. Intentar grabar. 3.- Comprobar que no se permite grabar y aparece un error de no tener permisos para realizar la acción.
RESULTADO ESPERADO	Requisito continúa en el estado en el que estuviese.

7.1 Cambio de estado de requisito padre a CERRADO por último hijo a CERRADO	
Validación de requisito	
RUTA DE PRUEBA	Abrir último requisito hijo a TESTEADO
OBJETIVO	Comprobar que si el requisito hijo que estamos pasando a CERRADO es el único que faltaba por pasar a CERRADO el padre pasa automáticamente también a CERRADO.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para pasar a CERRADO requisitos. 2.- Abrir un requisito TESTEADO hijo de otro que esté a CHEQUEADO, pasarlo a CERRADO, siendo este hijo el último que quedaba en un estado no terminal. 3.- Comprobar que tras grabar el requisito hijo, automáticamente el padre pasa a CERRADO.
RESULTADO ESPERADO	Requisito padre a CERRADO.

8.1 Cambio de estado de requisito padre por cambia estado hijos	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que si nos hemos llevado un requisito padre a CHEQUEADO ,si todos sus hijos son CERRADO, se pasa a CERRADO.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para modificar el requisito al estado concreto que se desee. 2.- Abrir requisito hijo último a un estado distinto de los demás hijos. 3.- Modificarlo al mismo estado que el resto de hermanos. 4.- Se modifica el estado del padre. 5.- Comprobar que todos los requisitos hijos de ponen a CERRADO.
RESULTADO ESPERADO	Requisitos padre e hijos al estado cambiado

8.2 Cambio de estado de requisito padre por cambia estado hijos	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que si nos hemos llevado un requisito padre a CHEQUEADO ,si todos sus hijos son CERRADO, se pasa a CERRADO.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para modificar el requisito al estado concreto que se desee. 2.- Abrir requisito hijo último a un estado distinto de los demás hijos. 3.- Modificarlo al mismo estado que el resto de hermanos. 4.- Se modifica el estado del padre. 5.- Comprobar que todos los requisitos hijos de ponen a CERRADO.
RESULTADO ESPERADO	Requisitos padre e hijos al estado cambiado

8.2 Requisito padre no se cambia a estado de hijos	
Validación de requisito	
RUTA DE PRUEBA	Abrir requisito existente
OBJETIVO	Comprobar que si el requisito hijo que estamos modificando su estado es el único que queda a un estado diferente, se analiza si van a quedar todos al mismo estado y el padre cambia a dicho estado sólo si se acepta el cambio. Comprobar que si no se acepta el cambio, no se modifica el requisito padre.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para modificar el requisito al estado concreto que se desee. 2.- Abrir requisito hijo último a un estado distinto de los demás hijos. 3.- Modificarlo al mismo estado que el resto de hermanos. Comprobar que se avisa de que se va a propagar el cambio de los hijos hacia arriba y se va a modificar el estado del padre para igualarlo. 4.- Comprobar que si se dice que no a dicho cambio, el padre se queda a su estado anterior y no se actualiza a los estados de los hijos.
RESULTADO ESPERADO	Requisito padre a estado distinto de hijos.

10.1 Mover requisito	
Validación de requisito	
RUTA DE PRUEBA	Vista jerárquica de herramienta
OBJETIVO	Comprobar que si el requisito hijo que estamos modificando su estado es el único que queda a un estado diferente, se analiza si van a quedar todos al mismo estado y el padre cambia a dicho estado sólo si se acepta el cambio. Comprobar que si no se acepta el cambio, no se modifica el requisito padre.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para modificar el requisito. 2.-Mover el requisito a otro requisito padre y comprobar que se puede grabar 3.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente)
RESULTADO ESPERADO	Requisito cambiado de sitio

10.2 Mover requisito no es posible sin permisos	
Caso de error	
RUTA DE PRUEBA	Vista jerárquica de herramienta
OBJETIVO	Comprobar que si el requisito hijo que estamos modificando su estado es el único que queda a un estado diferente, se analiza si van a quedar todos al mismo estado y el padre cambia a dicho estado sólo si se acepta el cambio. Comprobar que si no se acepta el cambio, no se modifica el requisito padre.
PROCESO	1.- Entrar a la aplicación con un usuario sin permisos para modificar el requisito. 2.- Mover el requisito a otro requisito padre y comprobar que nos aparece un mensaje de error ya que no se permite la modificación.
RESULTADO ESPERADO	Requisito en el mismo sitio.

11.1 Mover requisito de módulo de arquitectura	
Validación de requisito	
RUTA DE PRUEBA	Vista arquitectura de la herramienta
OBJETIVO	Comprobar que se puede cambiar un requisito en el árbol de componentes.
PROCESO	1.- Entrar a la aplicación con un usuario con permisos para modificar el requisito. 2.- Mover el requisito a otro componente y comprobar que se puede grabar correctamente. 3.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente)
RESULTADO ESPERADO	Requisito en otro módulo.

11.2 Mover requisito de componente no es posible sin permisos	
Caso de error	
RUTA DE PRUEBA	Vista arquitectura de herramienta
OBJETIVO	Comprobar que si no se tienen permisos no se puede cambiar un requisito en el árbol de arquitectura.
PROCESO	1.- Entrar a la aplicación con un usuario sin permisos para modificar el requisito. 2.- El estado del requisito es ABIERTO, CHEQUEADO ó ANALIZADO. 3.- Mover el requisito a otro componente y comprobar que nos aparece un mensaje de error y no se nos permite modificar el requisito de componente.
RESULTADO ESPERADO	Requisito en el mismo sitio.

11.3 Mover requisito de componente no es posible por estado	
Caso de error	
RUTA DE PRUEBA	Vista arquitectura de herramienta
OBJETIVO	Comprobar que si no se tiene el requisito en un estado concreto no se puede cambiar en el árbol de componentes.
PROCESO	1.- Entrar a la aplicación con un usuario sin permisos para modificar el requisito. 2.- El estado del requisito no es ABIERTO, CHEQUEADO ó ANALIZADO. 3.- Mover el requisito a otro componente y comprobar que nos aparece un mensaje de error y no se nos permite modificar el requisito de componente.
RESULTADO ESPERADO	Requisito en el mismo sitio.

12.1 Relacionar requisito con otro requisito	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que se pueden relacionar dos requisitos.
PROCESO	1.- Entrar a la aplicación con un usuario sin permisos para modificar el requisito. 2.- Abrir un requisito y después de buscar otro req añadirle una relación con otro. 3.- Comprobar que se graba la relación correctamente y que si abrimos el otro requisito podemos ver que existe la relación con éste. 4.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Requisito relacionado con otro.

12.2 Relacionar requisito con otro requisito no es posible al no tener permisos	
Caso de error	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar no que se pueden relacionar dos requisitos si no se tienen permisos.
PROCESO	1.- Entrar a la aplicación con un usuario sin permisos para modificar el requisito. 2.- Abrir un requisito y después de buscar otro req añadirle una relación con otro. 3.- Comprobar que nos da un error ya que no es posible relacionarlos porque no se tienen permisos.
RESULTADO ESPERADO	Requisito no se relacionado con otro.

13.1 Sugerir prioridad de implementación	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si eres un usuario de alto nivel se puede sugerir prioridad de implementación.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product o Project Manager. 2.- Abrir un requisito y en el cuadro de prioridades final, sugerir una prioridad de implementación. 3.- Comprobar que se puede grabar correctamente. 4.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Requisito con nueva prioridad de implementación sugerida.

13.2 Sugerir prioridad de implementación no es posible para todos usuarios	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si no eres un usuario de alto nivel no se puede sugerir prioridad de implementación.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product o Project Manager. 2.- Abrir un requisito y comprobar que no vemos el cuadro de prioridades de la pestaña Negotiate, con lo cual no podemos sugerir prioridad de implementación.
RESULTADO ESPERADO	Prioridad de implementación no sugerida.

14.1 Modificar prioridad sugerida	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si eres un usuario de alto nivel se puede sugerir prioridad de implementación.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product o Project Manager. 2.- Abrir un requisito y en el cuadro de prioridades final, sugerir una prioridad de implementación. 3.- Intentar modificar una prioridad de implementación que haya introducido dicho usuario. 4.- Comprobar que se puede grabar correctamente. Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Prioridad de implementación sugerida por dicho usuario modificada.

15.1 Fijar prioridad de desarrollo	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si eres Product Manager puedes fijar prioridad de desarrollo.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product Manager. 2.- Abrir un requisito e incluir prioridad de desarrollo. 3.- Comprobar que se puede grabar correctamente. 4.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Prioridad de desarrollo fijada.

15.2 Sugerir prioridad de desarrollo no es posible para todos usuarios	
Caso de error	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si no eres Product Manager no puedes fijar prioridad de desarrollo.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product Manager. 2.- Abrir un requisito e incluir prioridad de desarrollo. 3.- Comprobar que no se puede grabar ya que si no eres Product no puedes fijar prioridad de desarrollo.
RESULTADO ESPERADO	Prioridad de desarrollo no fijada.

15.3 Modificar prioridad de desarrollo	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que si no eres Product Manager no puedes fijar prioridad de desarrollo.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product Manager. 2.- Abrir un requisito y modifico prioridad de desarrollo. 3.- Comprobar que se puede grabar correctamente. 4.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Prioridad de implementación sugerida por dicho usuario modificada.

16.1 Asignar team a requisito	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que se pueden asignar participantes a requisito.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product o Project Manager. 2.- Abrir un requisito cualquiera y modificar su estado a CHEQUEADO. 3.- Comprobar que al grabar se piden los participantes 4.- Comprobar se graba el requisito.
RESULTADO ESPERADO	Participantes añadidos al requisito.

17.1 Modificar versión de desarrollo de un requisito	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que se puede modificar la versión de desarrollo de un requisito.
PROCESO	1.- Entrar en la aplicación siendo un usuario que sea Product Manager 2.- Abrir un requisito cualquiera y modificar su versión de desarrollo. 3.- Comprobar que se puede grabar correctamente. 4.- Comprobar también que se graba el cambio en el histórico del requisito(esto último no aplica actualmente).
RESULTADO ESPERADO	Versión de desarrollo del requisito modificada.

17.2 Modificar versión de desarrollo de un requisito no posible por permisos	
Validación de requisito	
RUTA DE PRUEBA	Abrir un requisito existente
OBJETIVO	Comprobar que no es posible añadir directorios de documentación al requisito.
PROCESO	1.- Entrar a la aplicación con un usuario cuyo role no sea Product Manager. 2.- Abrir un requisito cualquiera y modificar su versión de desarrollo. 3.- Comprobar que no se permite grabar y nos aparece un mensaje advirtiendo que si no eres product no puedes modificar la versión de desarrollo del requisito. 4.- Comprobar que si filtramos la vista jerárquica por la versión de desarrollo a la que hemos cambiado el requisito, lo veremos en la misma rama
RESULTADO ESPERADO	Versión de desarrollo del requisito no modificada.

18.1 Generar documento de requisito	
Validación de requisito	
RUTA DE PRUEBA	Desde vista principal de la aplicación
OBJETIVO	Comprobar que podemos llevar el contenido de un requisito a un documento.
PROCESO	1.- Desde la pantalla del requisito solicitar el documento del requisito. 2.- Comprobar que automáticamente nos genera dicho documento.
RESULTADO ESPERADO	Documento de requisito generado.

18.2 Generación fallida documento de requisito	
Caso de error	
RUTA DE PRUEBA	Desde vista principal de la aplicación
OBJETIVO	Comprobar que podemos llevar el contenido de un requisito a un documento.
PROCESO	1.- Desde la pantalla del requisito solicitar el documento del requisito. 2.- Comprobar que nos advierte con un mensaje de que no podemos genera el documento.
RESULTADO ESPERADO	Documento de requisito no generado.

19.1 Buscar un requisito	
Validación de requisito	
RUTA DE PRUEBA	Desde pantalla de Search
OBJETIVO	Comprobar que no es posible añadir directorios de documentación al requisito.
PROCESO	1.- En la utilidad de búsqueda de la herramienta introducimos algún criterio de búsqueda y comprobamos que tras buscar nos aparece el requisito o requisitos que cumplen dichas condiciones.
RESULTADO ESPERADO	Requisitos buscados.

20.1 Asociar directorios y no tener permisos	
Validación de requisito	
RUTA DE PRUEBA	Nuevo requisito o modificar requisito
OBJETIVO	Comprobar que no es posible añadir directorios de documentación al requisito.
PROCESO	1.- En la utilidad de búsqueda de la herramienta introducimos un usuario concreto y el role que ocupa en los requisitos que queremos buscar. 2.- Buscamos y veremos que aparecen los requisitos en los cuales el usuario tiene dicho role.
RESULTADO ESPERADO	Requisitos buscados.

21.1 Buscar por role y por estado	
Validación de requisito	
RUTA DE PRUEBA	Desde pantalla de Search
OBJETIVO	Comprobar que podemos buscar un requisito por el role del usuario y por el estado que
PROCESO	1.- En la utilidad de búsqueda de la herramienta introducimos un usuario concreto y el role que ocupa en los requisitos que queremos buscar. 2.- Buscamos y veremos que aparecen los requisitos en los cuales el usuario tiene dicho role y su estado es el indicado.
RESULTADO ESPERADO	Requisitos buscados.

22.1Cambiar vista por versión de desarrollo	
Validación de requisito	
RUTA DE PRUEBA	Desde vista jerárquica o de arquitectura
OBJETIVO	Comprobar que podemos cambiar la versión de desarrollo en cualquiera de las dos vistas.
PROCESO	1.- Desde la vista de jerarquía de requisitos o desde la de componentes, comprobar que podemos cambiar la versión de desarrollo y veremos los requisitos correspondientes a cada versión de desarrollo.
RESULTADO ESPERADO	Requisitos por versión de desarrollo en pantalla.

23.1 Activar vista de arquitectura	
Validación de requisito	
RUTA DE PRUEBA	Desde vista de arquitectura
OBJETIVO	Comprobar que podemos ver por vista jerárquica.
PROCESO	1.- Desde la vista de arquitectura de requisitos, comprobar que podemos cambiar vista jerárquica.
RESULTADO ESPERADO	Requisitos englobados en sus requisitos padres.

24.1 Activar vista jerárquica	
Validación de requisito	
RUTA DE PRUEBA	Desde vista de arquitectura
OBJETIVO	Comprobar que podemos ver por vista jerárquica.
PROCESO	1.- Desde la vista de arquitectura de requisitos, comprobar que podemos cambiar para poder ver por vista jerárquica. 2.- Abrir un requisito cualquiera ó al crear un requisito nuevo e introducir los directorios de documentación. 3.- Comprobar que al grabar el requisito nos aparece un mensaje de error que nos advierte de que no podemos modificar el mismo porque no tenemos permisos.
RESULTADO ESPERADO	Requisitos englobados en sus requisitos padres

7. FUTURAS LÍNEAS DE TRABAJO

7.1. CONSIDERACIONES GENERALES

La gestión de requisitos, y más globalmente la Ingeniería de Requisitos, y todas las actividades que conlleva, es una disciplina inexacta, y sin un referente absoluto. Esto quiere decir que no existe una forma, método que resulte indiscutible a la hora de llevar a cabo los procesos y tareas que forman parte de ella, y por tanto de la fase de Análisis. Cada organización, grupo de desarrolladores, equipos de proyectos, tienen una forma concreta de afrontar las labores de análisis y de por tanto gestionar los requerimientos que un sistema, aplicación o herramienta debe de cumplir para considerarse producto eficaz.

El Sistema Gestión de Requisitos es una alternativa a otros sistemas, y si bien es cierto que permite la gestión de requisitos, de una manera simplicista, y amigable para con el usuario final, es una herramienta que podría evolucionar en líneas futuras, y obviamente por tanto mejorar la gestión de los requisitos en algún sentido.

7.2. ARQUITECTURA

Una previsible futura línea de trabajo sobre el Sistema Gestión de Requisitos, sería la evolución del sistema, hacia un sistema con la máxima accesibilidad y movilidad, es decir, accesible desde cualquier lugar, dispositivo y momento. Para disponer de un sistema de este tipo sería necesario una sustitución del CLIENTE. En el Sistema Gestión de Requisitos, el cliente es una aplicación Windows, que debe ser instalada en cada máquina o terminal desde la que se quiera acceder al sistema. El terminal debe de disponer además de S.O Windows. En una futura línea de trabajo, el cliente dejaría de ser una aplicación de este tipo, y pasaría a ser un simple navegador, y, por tanto el sistema se transformaría en una aplicación web. Esto conllevaría grandes ventajas:

- Las aplicaciones web funcionan de manera similar independientemente del sistema operativo. VENTAJA: Accesible desde cualquier terminal o máquina que disponga de un navegador web.
- Las aplicaciones web en el caso de sufrir actualizaciones, no conllevan modificaciones en el cliente. VENTAJA: Facilidad de actualización y mantenimiento, no se necesita instalación en las maquinas o terminales.

- Las aplicaciones web requieren menos requisitos de memoria RAM de una máquina, es decir, tienen menos demanda de memoria que las aplicaciones instaladas localmente. VENTAJA: Dejan más espacio para la ejecución de otras aplicaciones en la máquina o terminal.

Queda claro por tanto que la línea futura del sistema sería el aprovechamiento de los servicios web actuales, dónde se concentra la lógica del negocio en servidor junto con la conexión a la base de datos, para el desarrollo de una aplicación web, realizada eso sí en un lenguaje de servidor (ASP.net sería quizá la opción más recomendable). Las clases de la lógica de negocio en cliente también serían aprovechables, por tanto, las modificaciones se concentrarían en la capa de presentación.

7.3. INTEGRACIÓN CON OTRAS HERRAMIENTAS

En futuras líneas de trabajo, una de las opciones que pudieran tenerse en cuenta sería la integración del sistema actual, con una herramienta que no sólo permita el seguimiento de los requisitos de un proyecto, sino del proyecto en general, permitiendo la creación de diagramas de Gantt, etc. en general sería interesante integrar la herramienta con un gestor de proyectos.

Otra herramienta interesante sería integrar en el sistema un foro, o el acceso a una red social, dónde se pudiera debatir online, sobre el estado de los requisitos, si un requisito debe ser desarrollado etc.

8. CONCLUSIONES

8.1. REFERENTES AL SISTEMA

En primer lugar, haciendo referencia al objetivo planteado al comienzo del desarrollo del Sistema Gestión de Requisitos, se ha conseguido desarrollar un sistema, capaz de resolver ineficiencias en la actividad de Gestión de Requisitos. Puedo afirmar que el hecho de hacer uso de una herramienta de estas características, en el desarrollo de esta aplicación, concretamente en la propia actividad de gestión de los requisitos, hubiera facilitado la labor del proceso de análisis y de la posterior evolución de los requisitos, en requisitos implementables. Se podría haber influido en el resultado final del proyecto, quizá reduciendo tiempos y como he citado antes acabando con alguna probable ineficiencia de otro tipo. En este sentido, se puede afirmar que se ha cumplido el objetivo del proyecto, debido a que el sistema creado permite conseguir lo citado anteriormente.

El Sistema Gestión de Requisitos, ha sido desarrollado hasta un punto, en el que las funcionalidades básicas para una gestión eficiente de la gestión de requisitos dentro de la Ingeniería de Requisitos, están cubiertas. Pero, bien es cierto, que el sistema, puede ser objeto de mejora y evolución, realizando una nueva iteración de desarrollo.

Si verdaderamente el uso del Sistema de Gestión de Requisitos es capaz de aportar valor añadido al trabajo dentro de departamentos de Desarrollo y de Calidad del Software, motivación para el desarrollo del mismo, solamente puede demostrarse implantándolo en los procesos de análisis de diferentes proyectos, y recibir *feedback* por parte de los usuarios del sistema.

Se debe valorar el uso de sistemas de este tipo, y he podido comprobar que su “no uso” puede implicar pérdida de calidad en el producto final desarrollado, en comparación a si se hacen uso de sistemas similares, debido, a que todo las actividades relacionadas con los requisitos resultan fundamentales en el proceso desarrollo de software, y un mal análisis y una mala especificación de requisitos, puede conllevar el fracaso de un proyecto, siendo ineficientes e ineficaces.

8.2. REFERENTES AL DESARROLLO DEL PROYECTO

Durante el tiempo que he trabajado en este proyecto la verdad es que he sacado múltiples conclusiones. Antes de describirlas, he de comentar, las dificultades iniciales para comenzar a desarrollarlo. Las ideas estaban claras, pero la forma de encarar el proyecto no estaba excesivamente clara. El proceso de análisis fue largo, más corto que el proceso de diseño, que gracias al buen producto obtenido (en mi opinión), de la fase de análisis provocó que se acortaran los tiempos esperados. Pero dónde más tiempo he dedicado, y volviendo a hacer referencia al periodo inicial, ha sido en la fase de aprendizaje de los conceptos relacionados con la plataforma sobre la cual se iba a implementar el sistema, es decir sobre la tecnología .Net. Fue un verdadero quebradero de cabeza empezar a desarrollar el sistema en relación a su fase de implementación.

Sin duda alguna, la realización de este proyecto, me ha permitido abordar áreas de conocimiento, que hasta ahora, conocía de su existencia, pero que durante los ya casi extintos estudios de Ingeniería Técnica en Informática de Gestión, no se abordaron. Pero también he de reconocer, que muchos de los conceptos y enseñanzas aprendidos durante los estudios, han vuelto a salir a flote en la realización del proyecto, y eso me ha permitido recordar aspectos fundamentales, por ejemplo, de la Ingeniería del Software, de la Gestión de Proyectos, y afianzar esos propios conocimientos, fundamentales para un ingeniero involucrado en el desarrollo de productos software.

Ha sido difícil concluir este proyecto, no por la temática del mismo, que me atraía por completo, pero sí por una serie de factores, entre ellos la dificulta de “arrancar” con la primera iteración de la implementación. Pero gracias a esta dificultad, también he de agradecer, que gracias a ese muro que parecía infranqueable, y al hecho de querer superarlo, actualmente y desde hace año y medio, trabajo desarrollando aplicaciones en la misma tecnología que el Sistema Gestión de Requisitos.

9. BIBLIOGRAFIA

9.1. MEMORIA DEL SISTEMA

1. Aplicaciones Informáticas de Gestión. Una perspectiva de Ingeniería del Software, Mario G. Piattini, José A.Calvo-Manzano, Joaquín Cervera, Luis Fernández, [2003] Ra-Ma
2. Software Engineering Economics, B.W Boehm, [1981]
3. Definición de Perfiles en Herramientas de Gestión de Requisitos, Dpto. de Lenguajes y Sistemas Informáticos e Ingeniería del Software UPM, Autor: Bárbara A.McDonald Landazuri, Profesor: Edmundo Tovar, [2005]
4. http://es.wikipedia.org/ Wikipedia, la enciclopedia libre
5. www.calidaddelsoftware.com Portal sobre calidad del software.
6. http://www.paper-review.com/tools/rms/read.php Comparativa de herramientas de gestión de requisitos del Mercado.
7. Ingeniería del Software: Un enfoque práctico, Roger S. Pressman, [1998] McGraw-Hill.
8. http://www.dcc.uchile.cl/~psalinas/uml/modelo.html Introducción muy sencilla en castellano al UML.
9. UML y Patrones. Una introducción al Análisis y Diseño orientado a objetos y al Proceso Unificado, Craig Larman, [2003] Prentice Hall
10. The consulting engineer, C. Maxwell Stanley, [1982]
11. http://www.visuresolutions.com/timon/Ingenieria%20de%20Requisitos_timon%20de%20los%20procesos%20de%20desarrollo%20de%20hoy%20en%20dia.pdf Ingeniería de requisitos

12. http://www.slideshare.net/juliopari/11-clase-analisis-de-requisitos Análisis de Requisitos

9.2. IMPLEMENTACIÓN DEL SISTEMA

13. http://www.c-sharpcorner.com/ Uno de los mejores *sites* sobre desarrollo en C#

14. http://www.elguille.info/ Una referencia en español.

15. http://www.josanguapo.com/ Desarrollo en C# y creación de servicios web.

16. http://www.vb-mundo.com/default.asp Desarrollo en varios lenguajes. Foros.